

Alphabet Size Reduction for Secure Network Coding: A Graph Theoretic Approach

Xuan Guang, *Member, IEEE*, and Raymond W. Yeung, *Fellow, IEEE*

Abstract

We consider a communication network where there exist wiretappers who can access a subset of channels, called a *wiretap set*, which is chosen from a given collection of wiretap sets. The collection of wiretap sets can be arbitrary. Secure network coding is applied to prevent the source information from being leaked to the wiretappers. In secure network coding, the required alphabet size is an open problem not only of theoretical interest but also of practical importance, because it is closely related to the implementation of such coding schemes in terms of computational complexity and storage requirement. In this paper, we develop a systematic graph-theoretic approach for improving Cai and Yeung's lower bound on the required alphabet size for the existence of secure network codes. The new lower bound thus obtained, which depends only on the network topology and the collection of wiretap sets, can be significantly smaller than Cai and Yeung's lower bound. A polynomial-time algorithm is devised for efficient computation of the new lower bound.

Index Terms

Information-theoretic security, secure network coding, wiretap network, alphabet size, lower bound, polynomial-time algorithm, graph theory.

I. INTRODUCTION

In Shannon's celebrated paper [1], the well-known *Shannon cipher system* is proposed, in which a sender wishes to transmit a private message to a receiver in the presence of a wiretapper, and it is required that the wiretapper can obtain no information about the message. For this purpose, the sender encrypts the message with a random key which is shared with the receiver via a "secure" channel and is inaccessible by the wiretapper. The encrypted message is transmitted to the receiver via a "public" channel which is eavesdropped by the wiretapper. The receiver can recover the message from the random

key and the encrypted message, while the wiretapper obtains no information about the message. In the literature, this is referred to as *information-theoretic security*.

Another well-known cipher system of information-theoretic security is *secret sharing*, proposed independently by Blakley [2] and Shamir [3], which is more elaborate than Shannon cipher system. In this system, a secret is encoded into shares which are distributed among a set of participants in such a way that only an arbitrarily specified qualified set of participants can recover the secret, while no information at all about the secret can be obtained from the shares of an unqualified set of participants.

In the context of communications, Ozarow and Wyner [4] proposed a related model called *wiretap channel II*. In this model, the sender's message is transmitted to the receiver through a set of noiseless point-to-point channels. It is assumed that a wiretapper can fully access any one but not more than one subset of the channels up to a certain size, which is referred to as a wiretap set. Logically, secret sharing contains wiretap channel II as a special case.

In 1978, Celebiler and Stette [5] proposed a scheme that can improve the efficiency of a two-way satellite communication system by performing the addition of two bits onboard the satellite. In 1999, Yeung and Zhang [6] studied the general coding problem in a satellite communication system and obtained an inner bound and an outer bound on the capacity region. In 2000, Ahlswede *et al.* [7] proposed the general concept of network coding that allows the intermediate nodes in a noiseless network to process the received information. In particular, they proved that if coding is applied at the nodes in a network, rather than routing only, the source node can multicast messages to all the sink nodes at the theoretically maximum rate, i.e., the smallest minimum cut capacity between the source node and a sink node, as the alphabet size of both the information source and the channel transmission symbol tends to infinity. Li *et al.* [8] further proved that linear network coding with a finite alphabet is sufficient for optimal multicast by means of a vector space approach. Independently, Koetter and Médard [9] developed an algebraic characterization of linear network coding by means of a matrix approach. The above two approaches correspond to the *global* and *local* descriptions of linear network coding, respectively. Jaggi *et al.* [10] proposed a deterministic polynomial-time algorithm for constructing a linear network code. In Tan *et al.* [11], the fundamental concept of linear independence among global encoding kernels was studied in depth. Based on this, a unified construction for different classes of linear network codes is obtained. It was shown explicitly in Sun *et al.* [12] that the linear independence structure of a *generic linear network code* naturally induces a matroid. An interesting characterization of the required field size of linear network codes over acyclic multicast networks was recently obtained by Sun *et al.* [13]. Their work reveals that the existence of a linear network code over a given finite field does not imply the existence of one over

all larger finite fields. For comprehensive discussions of network coding, we refer the reader to [14]–[18].

In the paradigm of network coding, information-theoretic security is naturally considered in the presence of a wiretapper. This problem, called *secure network coding*, was introduced by Cai and Yeung in [19], [20]. In the wiretap network model of secure network coding, the wiretapper, who can access any one wiretap set of edges, is not allowed to obtain any information about the private source message, while all the sink nodes as legal users can decode the private source message with zero error. Secret sharing can be formulated as a special case of secure network coding.

Similar to the coding for the classical wiretap models [1]–[4], in secure network coding, it is necessary to randomize the source message to guarantee information-theoretic security. El Rouayheb *et al.* [21] showed that the construction of secure network codes in [19], [20] can be viewed as a network generalization of the code construction for wiretap channel II in [4]. Motivated by El Rouayheb *et al.*, Silva and Kschischang [22] proposed a universal design of secure network codes via rank-metric codes such that the design of linear network codes for message transmission and the design of coding for security can be separated.

For secure network coding, the existing bound on the required alphabet size in [20]–[22] is roughly equal to the number of all wiretap sets, which is typically too large for implementation in terms of computational complexity and storage requirement. Therefore, the required alphabet size is a problem not only of theoretical interest but also of practical importance. Feldman *et al.* [23] showed that for a given security level, the alphabet size can be reduced by sacrificing a small fraction of the information rate. However, if the information rate is not sacrificed, even for the special case of an r -wiretap network, i.e., the wiretapper can access any one subset of at most r edges, whether it is possible to reduce the required alphabet size is not known [21]. Recently, for this special case, Guang *et al.* [24] proposed an equivalence relation of wiretap sets that can be applied to obtain an improved lower bound on the required field size. However, they did not provide any efficient algorithm for computing this bound.

In this paper, we fully explore the underlying mathematical structure of the approach in [24] and show that the required alphabet size for the existence of secure network codes can be reduced significantly, where the collection of the wiretap sets considered here is arbitrary. The main contributions and organization of the paper are given as follows:

- In Section II, we present secure network coding and the preliminaries, and introduce the necessary notation and definitions.
- In Section III, we generalize the equivalence relation amongst the wiretap sets in r -wiretap networks in [24] to general wiretap networks and introduce a domination relation amongst the equivalence

classes. We further prove that this domination relation is a strict partial order so that the set of the equivalence classes constitutes a strictly partially ordered set. The number of the maximal elements in this strictly partially ordered set is proved to be a lower bound on the required alphabet size, which in general is a significant improvement over the existing results. Our lower bound is applicable to both linear and non-linear secure network codes, and its improvement over the existing results can be unbounded.

- Our lower bound is graph-theoretical, and it depends only on the network topology and the collection of the wiretap sets. Section IV is devoted to the development of an efficient computation of our lower bound. Toward this end, we introduce the concept of *primary minimum cut*, by which we can bypass the complicated operations for determining the equivalence classes of wiretap sets and the domination relation among them. With this, a polynomial-time algorithm is developed for computing the lower bound.
- We conclude in Section V with a summary of our results and a remark on future research.

II. PRELIMINARIES

In this section, we first present the model of a wiretap network [19], [20] to be discussed in this paper. Let $G = (V, E)$ be a finite directed acyclic network with a single source node s and a set of sink nodes $T \subset V \setminus \{s\}$, where V and E are the sets of nodes and edges, respectively. In G , let $e = (u, v) \in E$ stand for a directed edge from node u to node v , where node u is called the *tail* of e and node v is called the *head* of e , denoted by $\text{tail}(e)$ and $\text{head}(e)$, respectively. Further, for a node v , define $\text{In}(v)$ as the set of incoming edges of v and $\text{Out}(v)$ as the set of outgoing edges of v . Formally, $\text{In}(v) = \{e \in E : \text{head}(e) = v\}$ and $\text{Out}(v) = \{e \in E : \text{tail}(e) = v\}$. Without loss of generality, assume $\text{In}(s) = \emptyset$ and $\text{Out}(t) = \emptyset$ for any sink node $t \in T$. An index taken from an alphabet can be transmitted on each edge e in E and parallel edges between two adjacent nodes are allowed. In other words, the capacity of each edge is taken to be 1. We make this assumption throughout the paper. Let \mathcal{A} be a collection of subsets of E , where every edge set in \mathcal{A} is called a *wiretap set*. Then a *wiretap network* is specified by a quadruple (G, s, T, \mathcal{A}) , where the source node s generates a source message and injects it into the network; each sink node $t \in T$ as a legal user is required to recover the source message with zero error; arbitrary one wiretap set in \mathcal{A} , but no more than one, may be fully accessed by a wiretapper. The collection \mathcal{A} of the wiretap sets is known by the source node and sink nodes but which wiretap set in \mathcal{A} is actually eavesdropped is unknown. Since the source node s and the sink node set T are usually fixed, we use (G, \mathcal{A}) to denote such a wiretap network for simplicity.

In a network G , if a sequence of edges (e_1, e_2, \dots, e_m) satisfies $\text{tail}(e_1) = u$, $\text{head}(e_m) = v$, and $\text{tail}(e_{k+1}) = \text{head}(e_k)$ for all $k = 1, 2, \dots, m-1$, we say that the sequence (e_1, e_2, \dots, e_m) is a path from node u (or edge e_1) to node v (or edge e_m). A *cut* between the source node s and a non-source node t is defined as a set of edges whose removal disconnects s from t . The *capacity* of a cut between s and t is defined as the number of edges in the cut, and the minimum of the capacities of all the cuts between s and t is called the *minimum cut capacity* between them. A cut between s and t is called a *minimum cut* if its capacity achieves the minimum cut capacity between them. These concepts can be extended to edge subsets of E . We first consider a cut between s and a set of non-source nodes T in the network G as follows. We create a new node t_T , and for every node t in T , add a new “super-edge” of infinite capacity¹ from t to t_T (which is equivalent to adding infinite parallel edges from t to t_T). A cut of the finite capacity between s and t_T is defined as a *cut* between s and T . We can naturally extend the capacity of a cut, the minimum cut capacity and the minimum cut to the case of T . Furthermore, let $A \subset E$ be an edge subset. Introduce a node t_e for each edge $e \in A$ which splits e into two edges e^1 and e^2 with $\text{tail}(e^1) = \text{tail}(e)$, $\text{head}(e^2) = \text{head}(e)$, and $\text{head}(e^1) = \text{tail}(e^2) = t_e$. Let $T_A = \{t_e : e \in A\}$ and then a cut between s and T_A is defined as a *cut* between s and A . In particular, if e^1 or e^2 appears in the cut, replace it by e . Similarly, the *minimum cut capacity* between s and A , denoted by $\text{mincut}(s, A)$, is defined as the minimum cut capacity between s and T_A , and a cut between s and A achieving the minimum cut capacity $\text{mincut}(s, A)$ is called a *minimum cut*. If an edge set $B \subseteq E$ is a cut between the source node s and a non-source node t (resp. a set of non-source nodes T and a set of edges A), then we say that the edge set B *separates* t (resp. T and A) from s . Note that if B separates t (resp. T and A) from s , then every path from s to t (resp. T and A) passes through at least one edge in B .

The following Menger’s theorem shows that the minimum cut capacity between node s to node t (resp. T and A) and the maximum number of edge-disjoint paths from s to t (resp. T and A) are really alternative ways to address the same issue.

Edge Version of Menger’s Theorem ([27, Theorem 6.7] and [28, Theorem 7.16]): The maximum number of edge-disjoint paths from node s to node t equals the minimum cut capacity between node s and node t .

In secure network coding, the source node s generates a random source message M according to an arbitrary distribution on a message set \mathcal{M} . The source message M is multicast to every sink node $t \in T$, while being protected from the wiretapper who can access any wiretap set A in \mathcal{A} . Similar to the other

¹Infinite symbols in the alphabet can be transmitted by one use of the edge.

information-theoretically secure models, in our wiretap network model, it is necessary to randomize the source message to combat the wiretapper. The randomness available at the source node, called the *key*, is a random variable K that takes values in a set of keys \mathcal{K} according to the uniform distribution.

Let \mathcal{F} be an alphabet. An \mathcal{F} -valued secure network code on a wiretap network (G, \mathcal{A}) consists of a set of local encoding mappings $\{\phi_e : e \in E\}$ such that for every e , ϕ_e is a mapping from $\mathcal{M} \times \mathcal{K}$ to the alphabet \mathcal{F} if $e \in \text{Out}(s)$, and is a mapping from $\mathcal{F}^{|\text{In}(v)|}$ to \mathcal{F} if $e \in \text{Out}(v)$ for a node $v \in V \setminus \{s\}$. The *information rate* of the secure network code is $\log_{|\mathcal{F}|} |\mathcal{M}|$.

To facilitate our discussion, let Y_e be the random variable transmitted on the edge e that is a function of the random source message M and the random key K . For a subset A of E , denote $(Y_e : e \in A)$ by Y_A .

Definition 1: For a secure network code on the wiretap network (G, \mathcal{A}) , $I(Y_A; M) = 0$ for every wiretap set $A \in \mathcal{A}$, where $I(Y_A; M)$ denotes the mutual information between Y_A and M .

The notion of security used in the definition of a secure network code is referred in the literature as *information-theoretic security* as oppose to *computational security*.

Proposition 1 ([20, Theorem 3]): Let (G, \mathcal{A}) be a wiretap network and \mathcal{F} be an alphabet with $|\mathcal{F}| \geq |T|$, the number of sink nodes in G . Then there exists an \mathcal{F} -valued secure network code over (G, \mathcal{A}) provided that $|\mathcal{F}| > |\mathcal{A}|$.²

If a wiretap set $A \in \mathcal{A}$ satisfies $|A| = \text{mincut}(s, A)$, then we say that the wiretap set A is *regular*. Further, if all wiretap sets A in \mathcal{A} are regular, then we say that the collection of wiretap sets \mathcal{A} is *regular*. For an arbitrary \mathcal{A} , by Proposition 1, there exists a secure network code if $|\mathcal{F}| > |\mathcal{A}|$. Now for each A in \mathcal{A} , replace it by a minimum cut CUT_A between s and A to form \mathcal{A}' . Observe that the minimum cut CUT_A is regular since

$$|\text{CUT}_A| = \text{mincut}(s, A) \leq \text{mincut}(s, \text{CUT}_A) \leq |\text{CUT}_A|,$$

where the first inequality follows from the fact that each cut separating CUT_A from s also separates A from s , and a secure network code which is secure for the wiretap sets in \mathcal{A}' is also secure for the wiretap sets in \mathcal{A} . Therefore, with respect to the bound given by Proposition 1, it suffices to consider regular wiretap sets. In the rest of the paper, we assume that all edge sets are regular unless otherwise specified.

In the following, we recall two concepts about strict and non-strict partial orders, which are used frequently in the paper.

²The reason for requiring $|\mathcal{F}| \geq |T|$ here is to guarantee the existence of a network code on G . In general, $|\mathcal{A}|$ is much larger than $|T|$.

Definition 2: Let \mathcal{D} be a finite set, and let “ $<$ ” and “ \leq ” be two binary relations amongst the elements in \mathcal{D} .

- The binary relation “ $<$ ” is called a *strict partial order* in \mathcal{D} if the following conditions are satisfied for arbitrary elements a , b , and c in \mathcal{D} :
 - 1) (**Irreflexivity**) $a \not< a$;
 - 2) (**Transitivity**) if $a < b$ and $b < c$, then $a < c$;
 - 3) (**Asymmetry**) if $a < b$, then $b \not< a$.³
- The binary relation “ \leq ” is called a *non-strict partial order* in \mathcal{D} if the following conditions are satisfied for arbitrary elements a , b , and c in \mathcal{D} :
 - 1) (**Reflexivity**) $a \leq a$;
 - 2) (**Antisymmetry**) if $a \leq b$ and $b \leq a$, then $a = b$;
 - 3) (**Transitivity**) if $a \leq b$ and $b \leq c$, then $a \leq c$.

III. REQUIRED ALPHABET SIZE FOR SECURE NETWORK CODING

In this section, for a wiretap network (G, \mathcal{A}) , we prove a new bound on the required alphabet size of the existence of secure network codes that improves upon the lower bound in [24]. In the next section, we present an efficient algorithm for evaluating this bound.

Let A and A' be two edge sets in G . Define a binary relation “ \sim ” between A and A' : $A \sim A'$ if and only if there exists an edge set CUT which is a minimum cut between s and A and also between s and A' , that is, A and A' have a common minimum cut between the source node s and each of them. Note that $A \sim A'$ implies $|A| = |A'|$ because $\text{mincut}(s, A) = |\text{CUT}| = \text{mincut}(s, A')$ and both A and A' are regular. It was proved in [24] that “ \sim ” is an equivalence relation. While reflexivity and symmetry of “ \sim ” are immediate, the proof of transitivity is nontrivial.

With the relation “ \sim ”, the wiretap sets in \mathcal{A} can be partitioned into equivalence classes. All the wiretap sets in an equivalence class have a common minimum cut, which is implied by the transitivity of “ \sim ”. To see this, consider wiretap sets A , A' , and A'' that are in the same equivalence class. Let A and A' have a common minimum cut CUT, and let A' and A'' have a common minimum cut CUT'. Then $\text{CUT} \sim A'$ and $\text{CUT}' \sim A'$. By the transitivity of “ \sim ”, we have $\text{CUT} \sim \text{CUT}'$, implying that there exists a common minimum cut between s and CUT and between s and CUT', which in turn is a minimum cut between s and A , between s and A' , and between s and A'' . Then we see by induction that all the wiretap sets

³Asymmetry can readily be deduced from irreflexivity and transitivity.

in the equivalence class can have a common minimum cut from s . Immediately, we give the following proposition.

Proposition 2: *Let A_1, A_2, \dots, A_m be m equivalent edge sets under the equivalence relation “ \sim ”. Then*

$$\text{mincut}(s, \cup_{i=1}^m A_i) = \text{mincut}(s, A_j), \quad \forall j, 1 \leq j \leq m. \quad (1)$$

Proof: Since the edge sets A_1, A_2, \dots, A_m are equivalent, they have a common minimum cut CUT separating each of them from s . This implies that

- 1) $|\text{CUT}| = \text{mincut}(s, A_j), \forall j, 1 \leq j \leq m;$
- 2) CUT is a cut between s and $\cup_{i=1}^m A_i$.

Combining 1) and 2), we have that for all $j, 1 \leq j \leq m,$

$$|\text{CUT}| = \text{mincut}(s, A_j) \leq \text{mincut}(s, \cup_{i=1}^m A_i) \leq |\text{CUT}|,$$

completing the proof. ■

Let $N(\mathcal{A})$ be the number of the equivalence classes in \mathcal{A} . According to Proposition 1 and the discussions that follow, by replacing each equivalence class of wiretap sets in \mathcal{A} by its common minimum cut, we see that there exists an \mathcal{F} -valued secure network code on (G, \mathcal{A}) provided that $|\mathcal{F}| > N(\mathcal{A})$. This lower bound $N(\mathcal{A})$ on $|\mathcal{F}|$ was originally obtained in [24] for r -wiretap networks, but it also applies for general wiretap networks. We use the following example to illustrate the advantage of this approach.

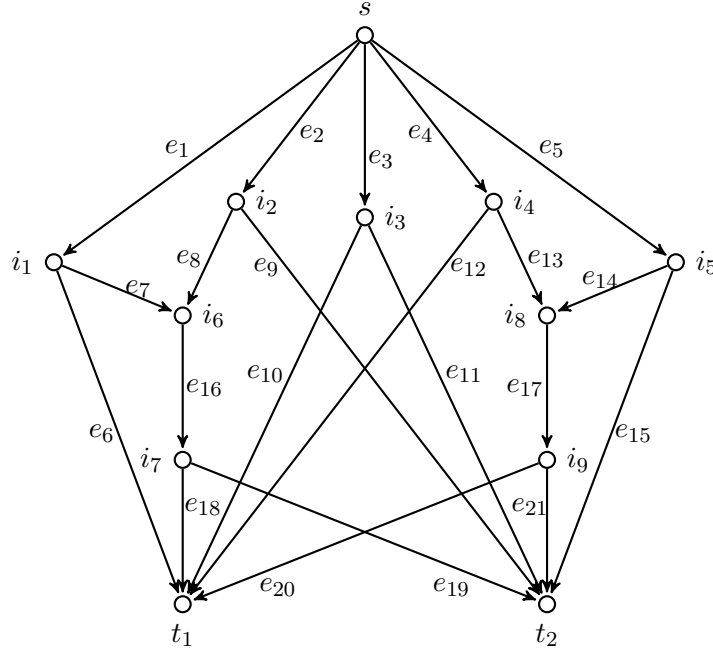


Fig. 1. The network G .

Example 1: Consider the network G depicted in Fig. 1. Let the collection of wiretap sets \mathcal{A} be

$$\begin{aligned} \mathcal{A} = & \left\{ \{e_6\}, \{e_7\}, \{e_8\}, \{e_9\}, \{e_{12}\}, \{e_{13}\}, \{e_{14}\}, \{e_{15}\}, \{e_{18}\}, \{e_{19}\}, \{e_{20}\}, \{e_{21}\}, \right. \\ & \{e_6, e_{18}\}, \{e_6, e_{19}\}, \{e_7, e_{18}\}, \{e_7, e_{19}\}, \{e_8, e_{11}\}, \{e_8, e_{16}\}, \{e_8, e_{18}\}, \{e_9, e_{10}\}, \\ & \{e_9, e_{18}\}, \{e_9, e_{19}\}, \{e_{10}, e_{14}\}, \{e_{10}, e_{15}\}, \{e_{10}, e_{19}\}, \{e_{10}, e_{21}\}, \{e_{11}, e_{14}\}, \{e_{11}, e_{15}\}, \\ & \{e_{11}, e_{18}\}, \{e_{11}, e_{20}\}, \{e_{12}, e_{20}\}, \{e_{12}, e_{21}\}, \{e_{13}, e_{17}\}, \{e_{13}, e_{21}\}, \{e_{14}, e_{20}\}, \{e_{14}, e_{21}\}, \\ & \{e_{15}, e_{20}\}, \{e_{15}, e_{21}\}, \{e_{18}, e_{20}\}, \{e_{18}, e_{21}\}, \{e_{19}, e_{20}\}, \{e_{19}, e_{21}\}, \\ & \left. \{e_1, e_3, e_{16}\}, \{e_1, e_{11}, e_{16}\}, \{e_2, e_{10}, e_{16}\}, \{e_3, e_5, e_{17}\}, \{e_4, e_{10}, e_{17}\}, \{e_5, e_{11}, e_{17}\} \right\}, \end{aligned}$$

with $|\mathcal{A}| = 48$. The equivalence classes of wiretap sets are

$$\begin{aligned} \text{Cl}_1 &= \left\{ \{e_6\}, \{e_7\} \right\}, & \text{Cl}_2 &= \left\{ \{e_8\}, \{e_9\} \right\}, & \text{Cl}_3 &= \left\{ \{e_{12}\}, \{e_{13}\} \right\}, \\ \text{Cl}_4 &= \left\{ \{e_{14}\}, \{e_{15}\} \right\}, & \text{Cl}_5 &= \left\{ \{e_{18}\}, \{e_{19}\} \right\}, & \text{Cl}_6 &= \left\{ \{e_{20}\}, \{e_{21}\} \right\}, \\ \text{Cl}_7 &= \left\{ \{e_8, e_{11}\}, \{e_9, e_{10}\} \right\}, & \text{Cl}_8 &= \left\{ \{e_{10}, e_{19}\}, \{e_{11}, e_{18}\} \right\}, & \text{Cl}_9 &= \left\{ \{e_{10}, e_{21}\}, \{e_{11}, e_{20}\} \right\}, \\ \text{Cl}_{10} &= \left\{ \{e_{10}, e_{14}\}, \{e_{10}, e_{15}\}, \{e_{11}, e_{14}\}, \{e_{11}, e_{15}\} \right\}, \\ \text{Cl}_{11} &= \left\{ \{e_{18}, e_{20}\}, \{e_{18}, e_{21}\}, \{e_{19}, e_{20}\}, \{e_{19}, e_{21}\} \right\}, \\ \text{Cl}_{12} &= \left\{ \{e_6, e_{18}\}, \{e_6, e_{19}\}, \{e_7, e_{18}\}, \{e_7, e_{19}\}, \{e_8, e_{16}\}, \{e_8, e_{18}\}, \{e_9, e_{18}\}, \{e_9, e_{19}\} \right\}, \\ \text{Cl}_{13} &= \left\{ \{e_{12}, e_{20}\}, \{e_{12}, e_{21}\}, \{e_{13}, e_{17}\}, \{e_{13}, e_{21}\}, \{e_{14}, e_{20}\}, \{e_{14}, e_{21}\}, \{e_{15}, e_{20}\}, \{e_{15}, e_{21}\} \right\}, \\ \text{Cl}_{14} &= \left\{ \{e_1, e_3, e_{16}\}, \{e_1, e_{11}, e_{16}\}, \{e_2, e_{10}, e_{16}\} \right\}, \\ \text{Cl}_{15} &= \left\{ \{e_3, e_5, e_{17}\}, \{e_4, e_{10}, e_{17}\}, \{e_5, e_{11}, e_{17}\} \right\}. \end{aligned}$$

Then $N(\mathcal{A}) = 15$, which is considerably smaller than $|\mathcal{A}|$.

However, [24] does not provide an algorithm for computing $N(\mathcal{A})$, making the bound practically not useful except for very simple networks for which $N(\mathcal{A})$ can be readily evaluated. This issue will be addressed in the next section after we have introduced the notion of equivalence-class domination in the rest of this section.

The equivalence class containing a wiretap set A is denoted by $\text{Cl}(A)$, or simply Cl if there is no ambiguity. Note that the wiretap sets have possibly different cardinalities, and a wiretap set may be separated from s by another wiretap set of a larger cardinality. If every wiretap set in an equivalence class can be separated by some wiretap set with a larger cardinality, then it is not necessary to consider this equivalence class for the purpose of lower bounding the required alphabet size. For instance in Example 1,

since both the wiretap sets $\{e_{18}\}$ and $\{e_{19}\}$ are separated by another wiretap set $\{e_1, e_3, e_{16}\}$, it is not necessary to consider $\text{Cl}_5 = \{\{e_{18}\}, \{e_{19}\}\}$. In the following, we explore the essence of this observation and establish in Theorem 5 a strict partial order amongst the equivalence classes, which can help further reduce the required alphabet size.

Definition 3 (Wiretap-Set Domination): Let A_1 and A_2 be two wiretap sets in \mathcal{A} with $|A_1| < |A_2|$. We say that A_1 is dominated by A_2 , denoted by $A_1 \prec A_2$, if there exists a minimum cut between s and A_2 that also separates A_1 from s . In other words, upon deleting the edges in the minimum cut between s and A_2 , s and A_1 are also disconnected.

Note that in the above definition, in order for $A_1 \prec A_2$, $|A_1|$ has to be *strictly smaller* than $|A_2|$, and $A_1 \prec A_2$ does not mean that A_2 is at the “upstream” of A_1 . For instance in Fig. 1, let $A_1 = \{e_3, e_8\}$ and $A_2 = \{e_6, e_{10}, e_{18}\}$. We have $A_2 \succ A_1$ since $\{e_1, e_2, e_3\}$ is a minimum cut between s and A_2 that separates A_1 from s , although A_1 is actually at the “upstream” of A_2 .

The following proposition gives a necessary and sufficient condition for the existence of a domination relation between two wiretap sets.

Proposition 3: For wiretap sets A_1 and A_2 such that $|A_1| < |A_2|$, $A_1 \prec A_2$ if and only if

$$\text{mincut}(s, A_1 \cup A_2) = \text{mincut}(s, A_2). \quad (2)$$

Proof: By Definition 3, the “only if” part is evident. We only need to prove the “if” part. Let CUT be a minimum cut separating $A_1 \cup A_2$ from s , so that by (2),

$$|\text{CUT}| = \text{mincut}(s, A_1 \cup A_2) = \text{mincut}(s, A_2),$$

which implies that CUT is also a minimum cut between s and A_2 . Since CUT is also a cut (not minimum because $|A_1| < |A_2|$) between s and A_1 , we have $A_1 \prec A_2$ by definition. This completes the proof. ■

We remark that although (1) for $m = 2$ is equivalent to (2), Proposition 2 and Proposition 3 are different because in Proposition 2, A_1, A_2, \dots, A_m have the same cardinality, while in Proposition 3 we have $|A_1| < |A_2|$.

Next, we extend the notion of domination to equivalence classes.

Definition 4 (Equivalence-Class Domination): For two distinct equivalence classes Cl_1 and Cl_2 , if there exists a common minimum cut of the wiretap sets in Cl_2 that separates all the wiretap sets in Cl_1 from s , we say that Cl_1 is dominated by Cl_2 , denoted by $\text{Cl}_1 \prec \text{Cl}_2$.⁴

⁴Here we use the same symbol “ \prec ” to represent two domination relations, but this abuse of notation should cause no ambiguity.

We also give a necessary and sufficient condition for the existence of a domination relation between two equivalence classes.

Theorem 4: *Let A_1 and A_2 be two wiretap sets in \mathcal{A} . Then $\text{Cl}(A_1) \prec \text{Cl}(A_2)$ if and only if $A'_1 \prec A'_2$ for all $A'_1 \in \text{Cl}(A_1)$ and $A'_2 \in \text{Cl}(A_2)$.*

Proof: See Appendix A. ■

For the equivalence-class domination relation, we have the following theorem.

Theorem 5: *The equivalence-class domination relation “ \prec ” amongst the equivalence classes in \mathcal{A} is a strict partial order.*

In order to prove Theorem 5, we need the following lemma. Denote by $\text{MinCut}(B)$ the set of the minimum cuts between s and an edge set B .

Lemma 6: *Let A_1 and A_2 be two wiretap sets and $A_1 \prec A_2$. Then for any $\text{CUT}_1 \in \text{MinCut}(A_1)$ and any $\text{CUT}_{1,2} \in \text{MinCut}(A_1 \cup A_2)$,*

$$\text{CUT}_1 \prec \text{CUT}_{1,2}.$$

Proof: See Appendix B. ■

An important consequence of Lemma 6 is the following theorem which enhances Theorem 4.

Theorem 7: *$\text{Cl}(A_1) \prec \text{Cl}(A_2)$ if and only if $A_1 \prec A_2$.*

Proof: By Definition 4, the “only if” part is obvious. In the following we prove the “if” part. Let CUT_i be a common minimum cut of the wiretap sets in $\text{Cl}(A_i)$, $i = 1, 2$. Let $\text{CUT}_{1,2} \in \text{MinCut}(A_1 \cup A_2)$, and so $\text{CUT}_{1,2} \sim A_2$ by Proposition 3.

Since $A_1 \prec A_2$, by Lemma 6 we have $\text{CUT}_1 \prec \text{CUT}_{1,2}$, i.e., there exists a minimum cut CUT between s and $\text{CUT}_{1,2}$ that separates CUT_1 from s . By Proposition 3, we further obtain

$$\text{mincut}(s, \text{CUT}_1 \cup \text{CUT}_{1,2}) = \text{mincut}(s, \text{CUT}_{1,2}),$$

implying that $\text{CUT} \sim \text{CUT}_{1,2} \sim A_2$. Together with $A_2 \sim \text{CUT}_2$, we obtain $\text{CUT} \sim \text{CUT}_2$. Thus, CUT and CUT_2 have a common minimum cut, denoted by CUT^* , which satisfies the following:

- 1) CUT^* is a common minimum cut between s and each of the wiretap sets in $\text{Cl}(A_2)$, since CUT^* is a minimum cut between s and CUT_2 .
- 2) CUT^* separates each of the wiretap sets in $\text{Cl}(A_1)$, since CUT^* is a minimum cut between s and CUT , and CUT separates CUT_1 from s .

It then follows by definition that $\text{Cl}(A_1) \prec \text{Cl}(A_2)$, completing the proof. ■

With lemma 6 and Theorem 7, we are now ready to prove Theorem 5.

Proof of Theorem 5: The irreflexivity can be easily proved by Definition 3 and Theorem 7 as follows. Assume $Cl \prec Cl$ for some equivalence class Cl , which implies by Theorem 7 that $A \prec A$ for any $A \in Cl$, a contradiction to the definition of wiretap-set domination (Definition 3).

To complete the proof, we only need to prove the transitivity of “ \prec ”, i.e., for three equivalence classes Cl_1 , Cl_2 , and Cl_3 , if $Cl_1 \prec Cl_2$ and $Cl_2 \prec Cl_3$, then $Cl_1 \prec Cl_3$. Let $A_i \in Cl_i$, $i = 1, 2, 3$. By Theorem 7 and Proposition 3, it is sufficient to prove that

$$\text{mincut}(s, A_1 \cup A_3) = \text{mincut}(s, A_3). \quad (3)$$

First, note that a cut separating $A_1 \cup A_3$ from s is also a cut between s and A_3 , which implies

$$\text{mincut}(s, A_1 \cup A_3) \geq \text{mincut}(s, A_3). \quad (4)$$

On the other hand, in light of $Cl_1 \prec Cl_2$ and $Cl_2 \prec Cl_3$, by Definition 4, there exists a common minimum cut CUT' of the wiretap sets in Cl_2 which separates all the wiretap sets in Cl_1 from s , and there exists a common minimum cut CUT'' of the wiretap sets in Cl_3 which separates all the wiretap sets in Cl_2 . Consequently, we have $CUT' \in \text{MinCut}(A_2)$ and $CUT'' \in \text{MinCut}(A_2 \cup A_3)$. In addition, we also have $A_2 \prec A_3$ by Theorem 7 since $Cl_2 \prec Cl_3$. Thus, it follows from Lemma 6 that

$$CUT' \prec CUT''. \quad (5)$$

Consider $CUT' \cup CUT''$, and note that a cut between s and $CUT' \cup CUT''$ separates $A_1 \cup A_2 \cup A_3$ from s . This implies

$$\text{mincut}(s, A_1 \cup A_3) \leq \text{mincut}(s, A_1 \cup A_2 \cup A_3) \leq \text{mincut}(s, CUT' \cup CUT''). \quad (6)$$

Together with (5) and Proposition 3, we further obtain that

$$\text{mincut}(s, CUT' \cup CUT'') = \text{mincut}(s, CUT''). \quad (7)$$

Since CUT'' is a common minimum cut of the wiretap sets in Cl_3 , we have

$$\text{mincut}(s, CUT'') = \text{mincut}(s, A_3). \quad (8)$$

By (6), (7), and (8), we obtain

$$\text{mincut}(s, A_1 \cup A_3) \leq \text{mincut}(s, A_3). \quad (9)$$

Then (3) follows from (4) and (9). The theorem is proved. ■

Since the set of all the equivalence classes in \mathcal{A} has been proved to be a strictly partially ordered set, we can define its *maximal equivalence classes* as follows.

Definition 5 (Maximal Equivalence Class⁵): For a collection of wiretap sets \mathcal{A} , an equivalence class Cl is a maximal equivalence class if there exists no other equivalence class Cl' such that $\text{Cl}' \succ \text{Cl}$. Denote by $N_{\max}(\mathcal{A})$ the number of the maximal equivalence classes in \mathcal{A} .

Let Cl be a maximal equivalence class and $\text{Cl}_1, \text{Cl}_2, \dots, \text{Cl}_m$ be m equivalence classes that are dominated by Cl . By Definition 4, for each $1 \leq i \leq m$, there exists a common minimum cut of the wiretap sets in Cl , denoted by CUT_i , that separates all the wiretap sets in Cl_i from s . For any wiretap set A in Cl , since each CUT_i is a common minimum cut of the wiretap sets in Cl , $\text{CUT}_i \sim A$. This implies that all CUT_i , $1 \leq i \leq m$, are equivalent by transitivity of the equivalence relation “ \sim ”. Using the argument immediately above Proposition 2, we see that CUT_i , $1 \leq i \leq m$, have a common minimum cut, say CUT . Then a secure network code which is secure for CUT is also secure for all the wiretap sets in every Cl_i , $1 \leq i \leq m$. Therefore, the number of maximal equivalence classes in \mathcal{A} gives a new lower bound on the required alphabet size, which is potentially an improvement over the lower bound $N(\mathcal{A})$.

Theorem 8: Let (G, \mathcal{A}) be a wiretap network and \mathcal{F} be the alphabet with $|\mathcal{F}| \geq |T|$, the number of sink nodes in G . Then there exists an \mathcal{F} -valued secure network code on (G, \mathcal{A}) provided that the alphabet size $|\mathcal{F}| > N_{\max}(\mathcal{A})$.

We continue to use the setup in Example 1 to illustrate the concepts mentioned above and the advantage of the new bound.

Example 2: Recall the wiretap network (G, \mathcal{A}) in Example 1. With the equivalence-class domination “ \prec ”, the strict partial order of the equivalence classes is illustrated by the Hasse diagram in Fig. 2, which shows that Cl_{11} , Cl_{14} , and Cl_{15} are all of the maximal equivalence classes, i.e., $N_{\max}(\mathcal{A}) = 3$, which is much smaller than $N(\mathcal{A}) = 15$.

In general, computing the values of $N(\mathcal{A})$ and $N_{\max}(\mathcal{A})$, or characterizing the corresponding Hasse diagram, is nontrivial. Even in the simple example, their values are not obvious. How to efficiently compute $N(\mathcal{A})$ and $N_{\max}(\mathcal{A})$ will be discussed in the next section.

It is easily seen that $N_{\max}(\mathcal{A}) \leq N(\mathcal{A}) \leq |\mathcal{A}|$, and in general $N_{\max}(\mathcal{A})$ can be much smaller than $|\mathcal{A}|$ as illustrated by Example 3 below. The only case when $N_{\max}(\mathcal{A})$ has no improvement over $|\mathcal{A}|$, i.e., $N_{\max}(\mathcal{A}) = |\mathcal{A}|$, is that every wiretap set itself forms an equivalence class and no domination relation exists amongst all the equivalence classes. In this case, the collection of wiretap sets \mathcal{A} is “sparse” and the value of $|\mathcal{A}|$ is already small.

⁵The maximal equivalence classes are those maximal elements in the set of equivalence classes, when this set is viewed as a strictly partially ordered set.

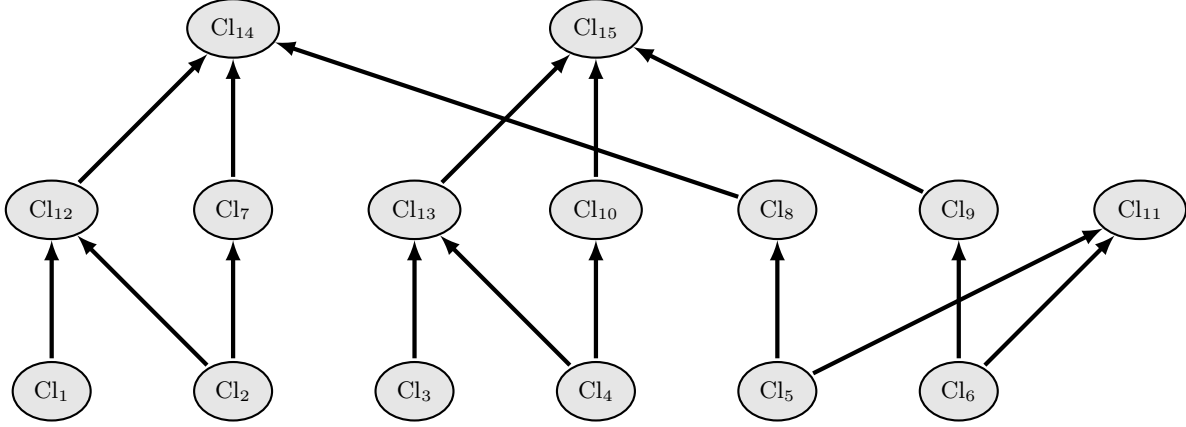


Fig. 2. The Hasse diagram of the set of all 15 equivalence classes, ordered by the equivalence-class domination relation “ \prec ”.

Example 3: Consider the combination network $G_{N,k}$ (see [14, p.26], [15, p.450]) with $N = 20$ and $k = 19$. In this network, there is a single source node s , 20 intermediate nodes, and $|T| = \binom{20}{19} = 20$ sink nodes. Each intermediate node is connected to s , and each sink node is connected to a distinct subset of 19 intermediate nodes. The number of the edges in the network is $|E| = N + |T| \cdot k = 400$, and the minimum cut capacity between s and every sink node is 19.

We partition all the edges into two layers: the upper and lower layers. The upper layer consists of $N = 20$ edges connecting the source node s and the intermediate nodes. The lower layer consists of $|T| \cdot k = 380$ edges connecting the intermediate nodes and the sink nodes. Assume that a wiretapper eavesdrops none of the edges in the upper layer and at most 17 edges in the lower layer, all of which are from distinct intermediate nodes. Note that the number of outgoing edges of each intermediate node is $\binom{N-1}{k-1} = 19$. Then the total number of wiretap sets is

$$|\mathcal{A}| = \sum_{i=1}^{17} \binom{20}{i} 19^i \gg 19^{17} \approx 5.48 \times 10^{21}.$$

Next, we compute $N(\mathcal{A})$ and $N_{\max}(\mathcal{A})$. Note that two wiretap sets A_1 and A_2 with $|A_1| = |A_2|$ are equivalent if and only if

$$\{\text{tail}(d) : d \in A_1\} = \{\text{tail}(e) : e \in A_2\}.$$

Then both A_1 and A_2 are dominated by the edge subset $\{d' = (s, \text{tail}(d)) : d \in A_1\}$. In general, for every wiretap set A in an equivalence class Cl , $\{\text{tail}(e) : e \in A\}$ are identical, and all the wiretap sets in Cl are dominated by the edge subset $\{e' = (s, \text{tail}(e)) : e \in A\}$. Thus, there is a one-to-one correspondence between an equivalence class and a subset of intermediate nodes, which has cardinality

no more than 17. Then $N(\mathcal{A}) = \sum_{i=1}^{17} \binom{20}{i} \approx 1.05 \times 10^6$, which is much smaller than $|\mathcal{A}|$.

Furthermore, an equivalence class Cl_1 is dominated by another one Cl_2 if and only if $\{\text{tail}(d) : d \in A_1\} \subsetneq \{\text{tail}(e) : e \in A_2\}$, where A_1 and A_2 are two arbitrary wiretap sets in Cl_1 and Cl_2 , respectively. In other words, $\text{Cl}_1 \prec \text{Cl}_2$ if and only if the subset of intermediate nodes corresponding to Cl_1 is strictly contained by the subset of intermediate nodes corresponding to Cl_2 . Thus, $N_{\max}(\mathcal{A}) = \binom{20}{17} = 1140$, which is in turn much smaller than $N(\mathcal{A})$.

In general, for a fixed k , the difference $N(\mathcal{A}) - N_{\max}(\mathcal{A}) \rightarrow \infty$ as $N \rightarrow \infty$. Therefore, the improvement of $N_{\max}(\mathcal{A})$ over $N(\mathcal{A})$ is unbounded.

IV. EFFICIENT ALGORITHM FOR COMPUTING THE LOWER BOUND

In Section III, a new lower bound on the required alphabet size of the existence of secure network codes over a wiretap network (G, \mathcal{A}) is obtained. This lower bound is graph-theoretical, and specifically it depends on the topology of the network G and the collection \mathcal{A} of wiretap sets. However, it is not given in a form which is readily computable. In this section, we develop a polynomial-time algorithm to compute this lower bound.

A. Primary Minimum Cut

Definition 6 (Primary Minimum Cut): Consider a finite directed acyclic network $G = (V, E)$ with a single source s , and let t be a non-source node in V . A minimum cut between s and t in G is primary, if it separates s and all the minimum cuts between s and t . In other words, a primary minimum cut between s and t is a common minimum cut of all the minimum cuts between s and t .

The notion of primary minimum cut is crucial to the development of our algorithm in the next subsection. We will first prove the existence and uniqueness of the primary minimum cut between the source node s and a non-source node t . In the following, we introduce the binary relation “ \leq ” amongst the minimum cuts between s and t .

Definition 7: Let CUT_1 and CUT_2 be two minimum cuts between s and t . We write $\text{CUT}_1 \leq \text{CUT}_2$, if CUT_1 separates CUT_2 from s , or equivalently, CUT_1 is a cut between s and CUT_2 .

This binary relation “ \leq ” between two minimum cuts is a non-strict partial order (Definition 2), as to be proved in the following theorem. This further implies that for two distinct minimum cuts CUT_1 and CUT_2 between s and t , $\text{CUT}_2 \not\leq \text{CUT}_1$ provided that $\text{CUT}_1 \leq \text{CUT}_2$.

Theorem 9: The binary relation “ \leq ” amongst the minimum cuts between s and t is a non-strict partial order.

Proof: Denote by $\text{MinCut}(t)$ the set of all minimum cuts between s and t , and let CUT_1 , CUT_2 , and CUT_3 be three minimum cuts in $\text{MinCut}(t)$. Reflexivity is apparent. For antisymmetry, we assume $\text{CUT}_1 \leq \text{CUT}_2$ and $\text{CUT}_2 \leq \text{CUT}_1$. By Definition 7, we obtain that CUT_1 (resp. CUT_2) separates CUT_2 (resp. CUT_1) from s . This implies $\text{CUT}_1 = \text{CUT}_2$.

To prove transitivity, i.e., if $\text{CUT}_1 \leq \text{CUT}_2$ and $\text{CUT}_2 \leq \text{CUT}_3$, then $\text{CUT}_1 \leq \text{CUT}_3$, we discuss the following two cases:

Case 1. At least two out of the three minimum cuts are the same. Transitivity is immediate.

Case 2. The three minimum cuts are distinct. Then, $\text{CUT}_1 \leq \text{CUT}_2$ and $\text{CUT}_2 \leq \text{CUT}_3$ mean that CUT_1 separates CUT_2 from s and CUT_2 separates CUT_3 from s , respectively. Consequently, CUT_1 separates CUT_3 from s , which proves $\text{CUT}_1 \leq \text{CUT}_3$ by Definition 7.

The theorem is proved. ■

The proof of the proposition below is straightforward and so it is omitted.

Proposition 10: *Let $n = \text{mincut}(s, t)$ and CUT be an arbitrary minimum cut between s and t . Then an arbitrary set of n edge-disjoint paths from s to t contains all the n edges in CUT , and each of the n edges is on an exactly one of the n edge-disjoint paths.*

For an acyclic network G , there exists an *upstream-to-downstream order* (also called *ancestral topological order*) on the edges in E , which is consistent with the natural partial order of the edges. To be specific, for two distinct edges d and e in E , if there is a directed path from d to e , we write $d \leq e$.⁶ We also set $e \leq e$, $\forall e \in E$. It is not difficult to see that the binary relation “ \leq ” amongst the edges in E is a non-strict partial order: the reflexivity and transitivity of “ \leq ” are immediate, and the antisymmetry follows from the acyclicity of G . The following lemma provides a necessary and sufficiency condition for $\text{CUT}_1 \leq \text{CUT}_2$, where CUT_1 and CUT_2 are two minimum cuts between s and t .

Lemma 11: *Let $\text{CUT}_1, \text{CUT}_2 \in \text{MinCut}(t)$, the set of all minimum cuts between s and t . Then $\text{CUT}_1 \leq \text{CUT}_2$ if and only if there exist $n = \text{mincut}(s, t)$ edge-disjoint paths P_1, P_2, \dots, P_n from s to t such that $e_{1,i} \leq e_{2,i}$, where $P_i \cap \text{CUT}_1 = \{e_{1,i}\}$ and $P_i \cap \text{CUT}_2 = \{e_{2,i}\}$ for all $1 \leq i \leq n$.*

Following Proposition 10 and Lemma 11, the next theorem asserts that the order between two minimum cuts under the relation “ \leq ” is independent of which set of n edge-disjoint paths from s to t is chosen. The proofs of Lemma 11 and Theorem 12 are relegated to Appendix C.

Theorem 12: *Let $n = \text{mincut}(s, t)$ and $\text{CUT}_1, \text{CUT}_2 \in \text{MinCut}(t)$ with $\text{CUT}_1 \leq \text{CUT}_2$, and P_1, P_2, \dots, P_n be n arbitrary edge-disjoint paths from s to t . Then $e_{1,i} \leq e_{2,i}$ holds for all $i = 1, 2, \dots, n$,*

⁶Here we use the same symbol “ \leq ” to represent two binary relations, but this abuse of notation should cause no ambiguity.

where $P_i \cap \text{CUT}_1 = \{e_{1,i}\}$ and $P_i \cap \text{CUT}_2 = \{e_{2,i}\}$, $1 \leq i \leq n$.

We now proceed to prove the existence and uniqueness of the primary minimum cut in Definition 6.

- 1) **Existence:** Let $\text{CUT}_1 = \{e_{1,i} : i = 1, 2, \dots, n\}$ and $\text{CUT}_2 = \{e_{2,i} : i = 1, 2, \dots, n\}$ be two minimum cuts in $\text{MinCut}(t)$, and P_1, P_2, \dots, P_n be n arbitrary edge-disjoint paths from s to t , where $n = \text{mincut}(s, t)$. We assume without loss of generality that $P_i \cap \text{CUT}_1 = \{e_{1,i}\}$ and $P_i \cap \text{CUT}_2 = \{e_{2,i}\}$ for $1 \leq i \leq n$. Define an edge set

$$\text{CUT} = \{e_i = \text{minord}(e_{1,i}, e_{2,i}) : i = 1, 2, \dots, n\},$$

where

$$\text{minord}(e_{1,i}, e_{2,i}) = \begin{cases} e_{1,i}, & \text{if } e_{1,i} \leq e_{2,i}; \\ e_{2,i}, & \text{otherwise.} \end{cases}$$

It was proved in [24, Lemma 5] that the above edge set CUT is also a minimum cut between s and t . By Lemma 11, we further have $\text{CUT} \leq \text{CUT}_1$ and $\text{CUT} \leq \text{CUT}_2$. Consequently, by Definition 7, CUT is a common minimum cut of CUT_1 and CUT_2 . Thus, we obtain $\text{CUT}_1 \sim \text{CUT}_2$ by the definition of the equivalence relation “ \sim ”. Similarly, we can prove that all the minimum cuts in $\text{MinCut}(t)$ are equivalent. We also have $|\text{MinCut}(t)| < \infty$ since the network G is finite. Thus, by the argument immediately before Proposition 2, there exists a common minimum cut CUT^* of all minimum cuts in $\text{MinCut}(t)$. In other words, CUT^* is a primary minimum cut between s and t by Definition 6.

- 2) **Uniqueness:** Let CUT_1^* and CUT_2^* be two primary minimum cuts between s and t . We obtain that $\text{CUT}_1^* \leq \text{CUT}_2^*$ and $\text{CUT}_2^* \leq \text{CUT}_1^*$ by Definitions 6 and 7. This implies $\text{CUT}_1^* = \text{CUT}_2^*$ by the antisymmetry of “ \leq ” in Theorem 9.

The concept of the primary minimum cut between the source node s and a non-source node t can be extended to between s and a wiretap set $A \in \mathcal{A}$ in the same way that the concept of a cut between s and t is extended to between s and A . In particular, for every wiretap set $A \in \mathcal{A}$, there exists a unique primary minimum cut between s and A , and further the minimum cut between s and the primary minimum cut is unique, i.e., itself.

Based on the above discussions, we now prove the next theorem, which shows that the computation of $N_{\max}(\mathcal{A})$ can be reduced to the computation of a set of primary minimum cuts such that each wiretap set $A \in \mathcal{A}$ (A is assumed to be regular) is separated from s by at least one primary minimum cut in this set. This theorem is the cornerstone in the development of our efficient algorithm to compute $N_{\max}(\mathcal{A})$.

Theorem 13: *Let A be a regular edge set in a finite directed acyclic network G with a single source node s , and CUT be the primary minimum cut between s and A . Then, the following hold:*

- 1) *For any regular edge set A' with $A' \sim A$, CUT is also the primary minimum cut between s and A' .*
- 2) *For any regular edge set B with $B \prec A$, CUT separates B from s .*

Proof: Let A' be a regular edge set with $A' \sim A$ and $A' \neq A$, and CUT' be the primary minimum cut between s and A' . We now prove that $\text{CUT} = \text{CUT}'$ as follows. First we see that $\text{CUT} \sim A$ and $\text{CUT}' \sim A'$. Together with $A \sim A'$, $\text{CUT} \sim \text{CUT}'$ follows from the transitivity of “ \sim ”. Thus, CUT and CUT' have a common minimum cut, denoted by CUT^* . While CUT^* separates CUT from s , CUT^* is also a minimum cut between s and A . Then it follows from Definition 7 that

$$\text{CUT}^* \leq \text{CUT}. \quad (10)$$

On the other hand, since CUT is the primary minimum cut between s and A , and CUT^* is a minimum cut between s and A , we also have $\text{CUT} \leq \text{CUT}^*$ by Definition 6. Combining this with (10), we obtain $\text{CUT} = \text{CUT}^*$. Similarly, we can prove that $\text{CUT}' = \text{CUT}^*$. Therefore, $\text{CUT} = \text{CUT}'$.

Next, we prove 2). Since $B \prec A$, by Definition 3, there exists a minimum cut CUT'' of A which separates B from s . Furthermore, since CUT is the primary minimum cut between s and A , we can see that $\text{CUT} \leq \text{CUT}''$ by Definitions 6 and 7. This implies that CUT separates B from s . The theorem is proved. ■

Corollary 14: *In a wiretap network (G, \mathcal{A}) , let Cl be an arbitrary equivalence class of the wiretap sets. Then*

- 1) *all the wiretap sets in Cl have the same primary minimum cut, which hence is called the primary minimum cut of the equivalence class Cl , and*
- 2) *for every equivalence class Cl' with $\text{Cl}' \prec \text{Cl}$, the primary minimum cut of Cl separates all the wiretap sets in Cl' from s .*

The above corollary can be proved by a straightforward application of Theorem 13. Since two maximal equivalence classes in (G, \mathcal{A}) cannot share a common primary minimum cut, this corollary shows that in order to compute $N_{\max}(\mathcal{A})$ for a wiretap network (G, \mathcal{A}) , it suffices to find the primary minimum cuts of all the maximal equivalence classes in (G, \mathcal{A}) .

B. Algorithm

Based on the observation at the end of the last subsection, we now develop an efficient algorithm for computing $N_{\max}(\mathcal{A})$. In our algorithm, the primary minimum cuts of all the maximal equivalence

classes in (G, \mathcal{A}) are obtained without first determining the equivalence classes of wiretap sets and the domination relation among them. This is the key to the efficiency of the algorithm. To be specific, we compute $N_{\max}(\mathcal{A})$ as follows:

- 1) Define a set \mathcal{B} , and initialize \mathcal{B} to the empty set.
- 2) Arbitrarily choose a wiretap set $A \in \mathcal{A}$ that has the largest cardinality in \mathcal{A} . Find the primary minimum cut between s and A , and call it CUT.
- 3) Partition the edge set E into two disjoint subsets: E_{CUT} and $E_{\text{CUT}}^c \triangleq E \setminus E_{\text{CUT}}$, where E_{CUT} is the set of the edges reachable from the source node s upon deleting the edges in CUT. Note that $\text{CUT} \subset E_{\text{CUT}}^c$.
- 4) Remove all the wiretap sets in \mathcal{A} that are subsets of E_{CUT}^c and add the primary minimum cut CUT to \mathcal{B} .
- 5) Repeat Steps 2) to 4) until \mathcal{A} is empty and output \mathcal{B} , where $N_{\max}(\mathcal{A}) = |\mathcal{B}|$.

The algorithm is explained as follows:

- In Step 2), since the algorithm always chooses a wiretap set $A \in \mathcal{A}$ that has the largest cardinality in \mathcal{A} , the chosen wiretap set A belongs to a maximal equivalence class.
- In Step 3), according to Corollary 14, the wiretap sets in the equivalence class $\text{Cl}(A)$ or an equivalence class Cl with $\text{Cl} \prec \text{Cl}(A)$ are subsets of E_{CUT}^c . Removing these wiretap sets from \mathcal{A} is equivalent to removing $\text{Cl}(A)$ and all equivalence classes Cl with $\text{Cl} \prec \text{Cl}(A)$.
- In addition, for any other equivalence class Cl' with $\text{Cl}' \not\prec \text{Cl}(A)$, by Theorem 7, we have $A' \not\prec A$ for any wiretap set $A' \in \text{Cl}'$. We now prove by contradiction that $A' \not\prec \text{CUT}$. Assume that $A' \prec \text{CUT}$. Then there exists a minimum cut CUT^* of CUT that separates s from A' . Since CUT is the primary minimum cut of A and CUT^* is a minimum cut of CUT, we have $\text{CUT}^* = \text{CUT}$ and CUT^* separates s from A . Then CUT^* is a (primary) minimum cut of A that separates s from A' , implying that $A' \prec A$, which is a contradiction to $A' \not\prec A$.
- As such, none of the wiretap sets in $\text{Cl}' \not\prec \text{Cl}(A)$ are removed from \mathcal{A} , and in particular, all maximal equivalence classes other than $\text{Cl}(A)$ are not removed from \mathcal{A} . Thus, exactly one maximal equivalence class is removed from \mathcal{A} in each iteration.

Remark 15: If in Step 4) we instead consider only those wiretap sets of the same cardinality as A , which means that only the wiretap sets in $\text{Cl}(A)$ are removed from \mathcal{A} , then the algorithm at the end outputs \mathcal{B} with $|\mathcal{B}| = N(\mathcal{A})$ instead of $N_{\max}(\mathcal{A})$.

In the proposed algorithm, we assume that all the wiretap sets in \mathcal{A} are regular. The algorithm can be modified so that it continues to be applicable without this assumption. This can be done by replacing “arbitrarily choose a wiretap set $A \in \mathcal{A}$ that has the largest cardinality in \mathcal{A} ” in Step 2) by “arbitrarily choose a wiretap set $A \in \mathcal{A}$ that has the largest minimum cut capacity in \mathcal{A} ”. However, this would require pre-computing the minimum cut capacity of every wiretap set in \mathcal{A} (this is essentially the same as replacing every non-regular wiretap set in \mathcal{A} by one of its minimum cuts, which is regular). Although the complexity for computing the minimum cut capacity of a wiretap set is only polynomial in $|E|$, this will still significantly increase the computational complexity of the algorithm when $|\mathcal{A}|$ is large. To avoid this shortcoming, we modify the original algorithm (which assumes that all the wiretap sets are regular) by replacing Step 4) by:

- 4') Remove all the wiretap or edge sets in $\mathcal{A} \cup \mathcal{B}$ that are subsets of E_{CUT}^c . Add the primary minimum cut CUT to \mathcal{B} .

An implementation of this algorithm can be found in Algorithm 1. Before we explain this algorithm, we first generalize two definitions and prove a lemma.

Definition 8: Two wiretap sets are equivalent if they have a common minimum cut.

Definition 9: Let A' and A be two wiretap sets in \mathcal{A} . Then A' is dominated by A (write $A' \prec A$) if $\text{mincut}(s, A') < \text{mincut}(s, A)$ and there exists a minimum cut between s and A that separates A' from s .

Definitions 8 and 9 are generalizations of their original versions that require the wiretap sets to be regular.

Lemma 16: For two wiretap sets A_1 and A_2 (not necessarily regular) in \mathcal{A} with $\text{mincut}(s, A_1) < \text{mincut}(s, A_2)$, $A_1 \prec A_2$ if and only if $\text{CUT}_{A_1} \prec \text{CUT}_{A_2}$, where CUT_{A_1} and CUT_{A_2} are the primary minimum cuts of A_1 and A_2 , respectively.

Proof: For the “if” part, since CUT_{A_2} is the primary minimum cut of A_2 and $\text{CUT}_{A_1} \prec \text{CUT}_{A_2}$, CUT_{A_2} separates CUT_{A_1} from s . Together with CUT_{A_1} being the (primary) minimum cut of A_1 , CUT_{A_2} separates A_1 from s , implying $A_1 \prec A_2$. For the “only if” part, since $A_1 \prec A_2$, there exists a minimum cut $\text{CUT}_{A_2}^*$ between s and A_2 that separates A_1 from s . This further implies that $\text{CUT}_{A_2}^*$ is also a minimum cut between s and $A_1 \cup A_2$. By Lemma 6, we have $\text{CUT}_{A_1} \prec \text{CUT}_{A_2}^*$. In addition, since CUT_{A_2} is the primary minimum cut of A_2 , we obtain $\text{CUT}_{A_2} \leq \text{CUT}_{A_2}^*$ by Definitions 6 and 7. Combining $\text{CUT}_{A_1} \prec \text{CUT}_{A_2}^*$ and $\text{CUT}_{A_2} \leq \text{CUT}_{A_2}^*$, we have proved that $\text{CUT}_{A_1} \prec \text{CUT}_{A_2}$. ■

We now explain Algorithm 1 as follows:

- In Step 4'), CUT_A is added to \mathcal{B} .

Algorithm 1: Algorithm for Computing $N_{\max}(\mathcal{A})$ **Input:** The wiretap network (G, \mathcal{A}) , where $G = (V, E)$.**Output:** $N_{\max}(\mathcal{A})$, the number of maximal equivalence classes with respect to (G, \mathcal{A}) .**begin**

```

1  Set  $\mathcal{B} = \emptyset$ ;
2  while  $\mathcal{A} \neq \emptyset$  do
3      choose a wiretap set  $A$  of the largest cardinality in  $\mathcal{A}$ ;
4      find the primary minimum cut CUT of  $A$ ;
5      partition  $E$  into two parts  $E_{\text{CUT}}$  and  $E_{\text{CUT}}^c = E \setminus E_{\text{CUT}}$ ;
6      for each  $B \in \mathcal{A} \cup \mathcal{B}$  do
7          if  $B \subseteq E_{\text{CUT}}^c$  then
8              remove  $B$  from  $\mathcal{A}$ .
9          end
6      end
9      add CUT to  $\mathcal{B}$ .
      end
10 Return  $\mathcal{B}$ .                                     // Note that  $|\mathcal{B}| = N_{\max}(\mathcal{A})$ .
end

```

- If A has the largest minimum cut capacity in \mathcal{A} , then CUT_A will stay in \mathcal{B} until the algorithm terminates. This implies that A belongs to a maximal equivalent class. Otherwise, there must exist a wiretap set A' in \mathcal{A} such that $A' \succ A$. By Definition 9, this implies that $\text{mincut}(s, A') > \text{mincut}(s, A)$, contradicting the fact that A has the largest minimum cut capacity in \mathcal{A} .
- If A does not have the largest minimum cut capacity in \mathcal{A} ,
 - 1) if A belongs to a maximal equivalence class (e.g. Cl_{11} in Example 2), then by Lemma 16, CUT_A will stay in \mathcal{B} until the algorithm terminates;
 - 2) otherwise (e.g. Cl_{12} in Example 2), by Lemma 16, CUT_A will subsequently be replaced by some primary minimum cut $\text{CUT}_{A''}$ of a wiretap set A'' in \mathcal{A} . Repeat this argument if necessary until a primary minimum cut CUT_{A^*} is added to \mathcal{B} , where $\text{Cl}(A^*)$ is a maximal equivalence class such that $A^* \succ A$.

- Combining all the above, we see that at the end the algorithm outputs \mathcal{B} that contains all the primary minimum cuts of the maximal equivalence classes in \mathcal{A} , and computes the minimum cut capacity of a wiretap set for at most $N(\mathcal{A})$ times (instead of $|\mathcal{A}|$ times).

In Algorithm 1, two key steps, namely finding the primary minimum cut and the edge partition (Lines 4 and 5 in Algorithm 1, respectively), are involved. The edge partition can be implemented efficiently by slightly modifying existing search algorithms on directed graphs [27], [28]. We can use a classical search algorithm to find all the nodes reachable along directed paths from the source node s . To find all the edges in E_{CUT} , i.e., the edges reachable from s upon deleting the edges in CUT, it suffices to add a simple functionality for storing the reachable edges during the search process. The implementation is given in Algorithm 2. In [27], it is shown that the search algorithm runs in $\mathcal{O}(|E|)$ time because in the worst case the algorithm needs to traverse all the edges in E . Here, since the primary minimum cut CUT of the wiretap set A is removed from the network G , Algorithm 2 can find the edge set E_{CUT} in $\mathcal{O}(|E_{\text{CUT}}|)$ time.

Before giving an efficient algorithm for finding the primary minimum cut, we first introduce some notation below. Let $G = (V, E)$ be a directed acyclic network with a single source node s and t be a sink node in $V \setminus \{s\}$. Denote by C_t the minimum cut capacity between the source node s and the sink node t . By the max-flow min-cut theorem [25], [26], the value $v(f)$ of a maximum flow f from s to t is equal to the minimum cut capacity C_t between s and t , i.e., $v(f) = C_t$. Since all the edges in the network G have unit-capacity (i.e., the capacity is 1), C_t is a positive integer and the maximum flow f can be decomposed into C_t edge-disjoint paths from s to t . Various algorithms for finding such edge-disjoint paths can be implemented in polynomial time in $|E|$ [27], [28].

Now, we explore efficient algorithms for finding the primary minimum cut between s and an edge set. For the convenience of presentation, we instead consider algorithms for finding the primary minimum cut between s and a node $t \neq s$. Toward this end, we propose Algorithm 3 which takes as input a set of C_t edge-disjoint paths from s to t . Such a set of paths can be obtained by using any existing algorithm for this purpose. The verification of Algorithm 3 is given in Appendix D. We give an example below to illustrate the algorithm.

Example 4: A directed acyclic network G with a maximum flow f from s to t is depicted in Fig. 3(a), where s and t are the source node and the sink node, respectively. Fig. 3 illustrates Algorithm 3 that outputs the primary minimum cut between s and t in G :

- At first, only the pair of nodes (s, i_3) satisfies Line 3 in Algorithm 3, i.e., \exists a forward edge from s to i_3 with the flow value 0. Update S to $\{s, i_3\}$. Next, for i_3 , (i_3, i_7) and (i_3, i_9) are two pairs of

Algorithm 2: Search Algorithm

```

begin
1  Unmark all nodes in  $V$ ;
2  mark source node  $s$ ;
3   $\text{pred}(s) := 0$ ;                                //  $\text{pred}(i)$  refers to a predecessor node of node  $i$ .
4  set the edge-set  $\text{SET} = \emptyset$ ;
5  set the node-set  $\text{LIST} = \{s\}$ ;
6  while  $\text{LIST} \neq \emptyset$  do
7      select a node  $i$  in  $\text{LIST}$ ;
8      if node  $i$  is incident to an edge  $(i, j)$  such that node  $j$  is unmarked then
9          mark node  $j$ ;
10          $\text{pred}(j) := i$ ;
11         add node  $j$  to  $\text{LIST}$ ;
12         add all parallel edges leading from  $i$  to  $j$  to  $\text{SET}$ ;
13     else
14         delete node  $i$  from  $\text{LIST}$ ;
15     end
16 end
17 Return the edge-set  $\text{SET}$ .
end

```

nodes such that there exist two forward edges with the flow value 0 from i_3 to i_7 and i_9 , respectively.

Update S to $\{s, i_3, i_7, i_9\}$. This is illustrated in Fig. 3(b).

- For $i_7 \in S$, the edge (i_7, i_{10}) is the only edge connecting i_7 with another node in $V \setminus S$. It is a forward edge but with the flow value 1 and does not satisfy Line 3 in Algorithm 3. For i_9 , since $i_9 \in S$, $i_6 \notin S$ and (i_9, i_6) is a reverse edge with the flow value 1, update S to $\{s, i_3, i_6, i_7, i_9\}$. Similarly, we further update S to $\{s, i_2, i_3, i_5, i_6, i_7, i_9\}$ by considering $i_6 \in S$. This is illustrated in Fig. 3(c).
- Finally, since (i_5, i_1) is a reverse edge with the flow value 1, we obtain $S = \{s, i_1, i_2, i_3, i_5, i_6, i_7, i_9\}$, as illustrated in Fig. 3(d).

Algorithm 3: Algorithm for Finding the Primary Minimum Cut

Input: An acyclic network $G = (V, E)$ with a maximal flow f from the source node s to a sink node t , i.e., for every edge e in the corresponding C_t edge-disjoint paths, the flow value is defined as 1, written as $f(e) = 1$; otherwise, the flow value is defined as 0, written as $f(e) = 0$.

Output: The primary minimum cut between s and t .

```

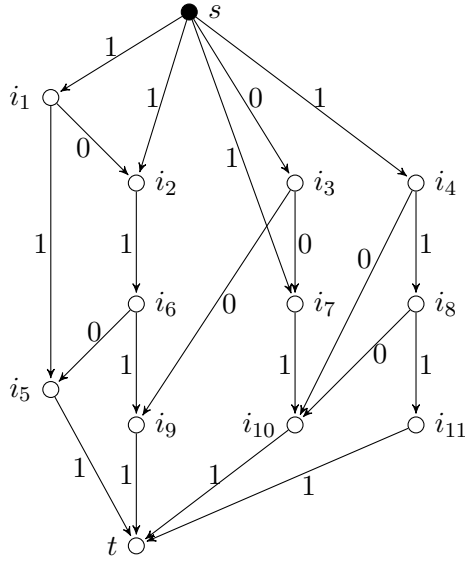
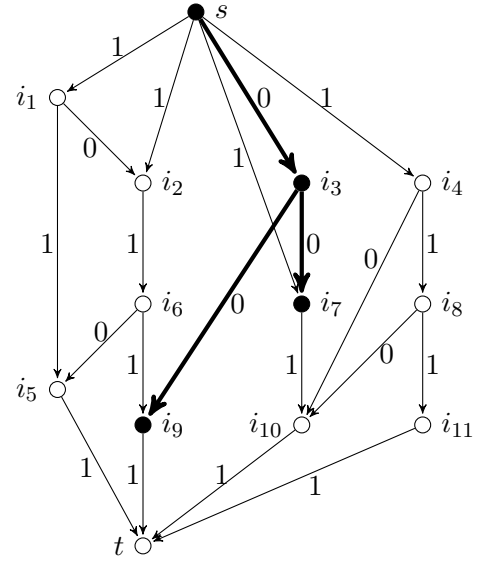
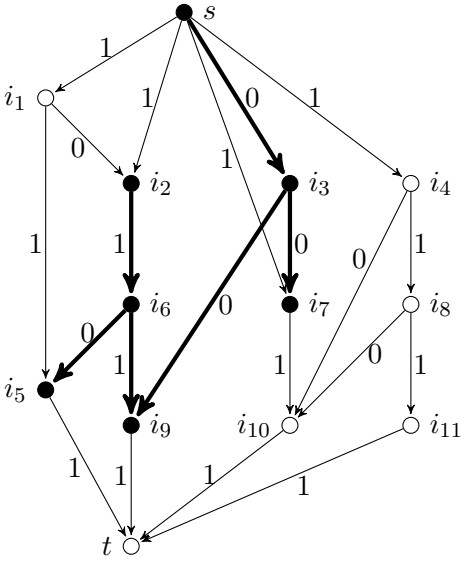
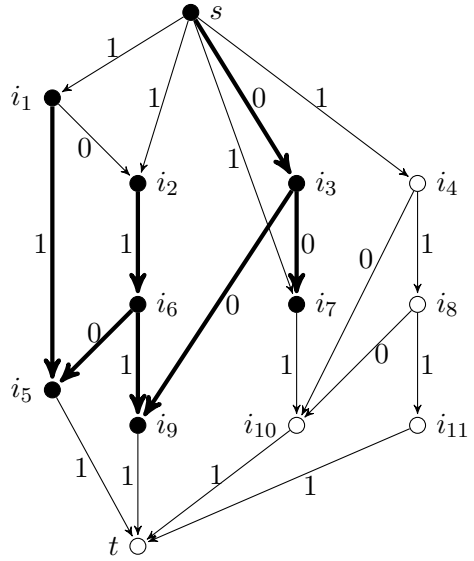
begin
1  Set  $S = \{s\}$ ;
2  for each node  $i \in S$  do
3      if  $\exists$  a node  $j \in V \setminus S$  s.t. either  $\exists$  a forward edge  $e$  from  $i$  to  $j$  s.t.  $f(e) = 0$  or  $\exists$  a reverse
        edge  $e$  from  $j$  to  $i$  s.t.  $f(e) = 1$  then
4          replace  $S$  by  $S \cup \{j\}$ .
        end
      end
5  Return  $\text{CUT} = \{e : \text{tail}(e) \in S \text{ and } \text{head}(e) \in V \setminus S\}$ .
end

```

Then the output edge set that is the primary minimum cut between s and t on G is

$$\text{CUT} = \{e : \text{tail}(e) \in S \text{ and } \text{head}(e) \in V \setminus S\} = \{(i_5, t), (i_9, t), (i_7, i_{10}), (s, i_4)\}.$$

In fact, Algorithm 3 can be regarded as the last part of the augmenting path algorithm [25], [26] (also see [27, Chapter 6.5] and [28, Chapter 7.2]) for determining the termination of the algorithm, i.e., the flow value cannot be further increased. Algorithm 3 requires at most $\mathcal{O}(|E|)$ time since the search method examines each edge at most once. If we use the augmenting path algorithm to find C_t edge-disjoint paths from s to t , then Algorithm 3 is already incorporated, and the total computational complexity for finding the primary minimum cut between s and t is at most $\mathcal{O}(C_t \cdot |E|)$ since the path augmentation approach requires at most $\mathcal{O}(|E|)$ time and the number of the path augmentations is upper bounded by the minimum cut capacity C_t . This total computational complexity may be reduced by employing more efficient maximum-flow algorithms for finding C_t edge-disjoint paths from s to t . For instance, if we suitably combine the features of the augmenting path algorithms and the shortest augmenting path algorithms [29], [30], the total computational complexity is $\mathcal{O}(\min\{C_t^{2/3}|E|, |E|^{3/2}\})$, which is better

(a) Initialize S to $\{s\}$.(b) Update S to $\{s, i_3, i_7, i_9\}$.(c) Update S to $\{s, i_2, i_3, i_5, i_6, i_7, i_9\}$.(d) Update S to $\{s, i_1, i_2, i_3, i_5, i_6, i_7, i_9\}$.Fig. 3. An example to use Algorithm 3 for finding the primary minimum cut between s and t on the network G .

than $\mathcal{O}(C_t \cdot |E|)$.

Next, we continue to use the setup in Examples 1 and 2 to illustrate Algorithm 1 for computing $N_{\max}(\mathcal{A})$.

Example 5: Recall the wiretap network (G, \mathcal{A}) in Examples 1 and 2. Define a set \mathcal{B} and initialize

\mathcal{B} to the empty set.

Step 1: Arbitrarily choose a wiretap set A_1 in \mathcal{A} of the largest cardinality 3, for instance, $A_1 = \{e_1, e_{11}, e_{16}\}$. Find $\text{CUT}_{A_1} = \{e_1, e_2, e_3\}$, the primary minimum cut between s and A_1 , by Algorithm 3. By Algorithm 2, we obtain the edge set

$$E_{\text{CUT}_{A_1}}^c = \{e_1, e_2, e_3, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{16}, e_{18}, e_{19}\}.$$

Remove the wiretap sets from \mathcal{A} that are subsets of $E_{\text{CUT}_{A_1}}^c$, e.g., $\{e_6\}$, $\{e_8, e_{18}\}$, $\{e_1, e_{11}, e_{16}\}$, etc. Update \mathcal{A} to

$$\begin{aligned} & \left\{ \{e_{12}\}, \{e_{13}\}, \{e_{14}\}, \{e_{15}\}, \{e_{20}\}, \{e_{21}\}, \{e_{10}, e_{14}\}, \{e_{10}, e_{15}\}, \{e_{10}, e_{21}\}, \right. \\ & \{e_{11}, e_{14}\}, \{e_{11}, e_{15}\}, \{e_{11}, e_{20}\}, \{e_{12}, e_{20}\}, \{e_{12}, e_{21}\}, \{e_{13}, e_{17}\}, \{e_{13}, e_{21}\}, \\ & \{e_{14}, e_{20}\}, \{e_{14}, e_{21}\}, \{e_{15}, e_{20}\}, \{e_{15}, e_{21}\}, \{e_{18}, e_{20}\}, \{e_{18}, e_{21}\}, \{e_{19}, e_{20}\}, \\ & \left. \{e_{19}, e_{21}\}, \{e_3, e_5, e_{17}\}, \{e_4, e_{10}, e_{17}\}, \{e_5, e_{11}, e_{17}\} \right\}. \end{aligned}$$

We remark that every wiretap set in the updated \mathcal{A} has at least one edge not in $E_{\text{CUT}_{A_1}}^c$. Add CUT_{A_1} to \mathcal{B} so that \mathcal{B} becomes $\{\text{CUT}_{A_1}\}$;

Step 2: Arbitrarily choose a wiretap set A_2 in the updated \mathcal{A} of the largest cardinality 3, say $A_2 = \{e_4, e_{10}, e_{17}\}$, and then find its primary minimum cut $\text{CUT}_{A_2} = \{e_3, e_4, e_5\}$ by Algorithm 3. Then use Algorithm 2 to obtain the edge set

$$E_{\text{CUT}_{A_2}}^c = \{e_3, e_4, e_5, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{17}, e_{20}, e_{21}\},$$

and remove the wiretap sets from \mathcal{A} that are subsets of $E_{\text{CUT}_{A_2}}^c$. Update \mathcal{A} to

$$\left\{ \{e_{18}, e_{20}\}, \{e_{18}, e_{21}\}, \{e_{19}, e_{20}\}, \{e_{19}, e_{21}\} \right\},$$

and add CUT_{A_2} to \mathcal{B} so that \mathcal{B} becomes $\{\text{CUT}_{A_1}, \text{CUT}_{A_2}\}$;

Step 3: Arbitrarily choose a wiretap set $A_3 = \{e_{19}, e_{20}\}$ in the updated \mathcal{A} of the largest cardinality 2 and find its primary minimum cut $\text{CUT}_{A_3} = \{e_{16}, e_{17}\}$. All the wiretap sets in \mathcal{A} are subsets of $E_{\text{CUT}_{A_3}}^c = \{e_{16}, e_{17}, e_{18}, e_{19}, e_{20}, e_{21}\}$. Then update \mathcal{A} to the empty set and add CUT_{A_3} to \mathcal{B} so that \mathcal{B} becomes $\{\text{CUT}_{A_1}, \text{CUT}_{A_2}, \text{CUT}_{A_3}\}$.

Algorithm 1 terminates and outputs \mathcal{B} . Then we have $N_{\max}(\mathcal{A}) = |\mathcal{B}| = 3$.

V. CONCLUSIONS

In this paper, we have proved a new lower bound on the required alphabet size for the existence of secure network codes. Our lower bound depends only on the network topology and the collection of the wiretap sets. Our result shows that in general the required alphabet size can be reduced significantly without sacrificing security and information rate. Since our bound is not in closed form, we also have proposed a polynomial-time algorithm to compute it efficiently.

Toward developing our lower bound and the efficient algorithm for computing this bound, we have introduced/discussed various graph-theoretic concepts, including the equivalence relation between two edge sets (first appeared in [24]), the domination relation among equivalence classes of edge sets, the primary minimum cut between the source node and a sink node, etc. Although in this paper these concepts are applied solely in the context of secure network coding, they appear to be of fundamental interest in graph theory and we expect that they will find applications in graph theory and beyond.

APPENDIX A

PROOF OF THEOREM 4

First assume $\text{Cl}(A_1) \prec \text{Cl}(A_2)$. By Definition 4, there exists a common minimum cut CUT of the wiretap sets in $\text{Cl}(A_2)$ that also separates each of the wiretap sets in $\text{Cl}(A_1)$ from s . For any $A'_1 \in \text{Cl}(A_1)$ and $A'_2 \in \text{Cl}(A_2)$, we have

$$|A_2| = \text{mincut}(s, A'_2) \leq \text{mincut}(s, A'_1 \cup A'_2) \leq |\text{CUT}| = |A_2|,$$

where the first and the last equalities follows from $A_2 \sim A'_2$ and $A_2 \sim \text{CUT}$, respectively. Together with Proposition 3, this proves the “only if” part.

We next prove the “if” part. Assume that $A'_1 \prec A'_2$ for any $A'_1 \in \text{Cl}(A_1)$ and $A'_2 \in \text{Cl}(A_2)$. For any $A'_1 \in \text{Cl}(A_1)$, since a minimum cut between s and $A'_1 \cup A_2$ is a cut between s and A_2 , together with the condition $A'_1 \prec A_2$ and Proposition 3, it follows that

$$\text{mincut}(s, A_2) = \text{mincut}(s, A'_1 \cup A_2), \quad (11)$$

i.e., a minimum cut between s and $A'_1 \cup A_2$ is actually a minimum cut between s and A_2 . Define $\text{MinCut}(A'_1 \cup A_2)$ as the set of all the minimum cuts between s and $A'_1 \cup A_2$. Then for any $A'_1 \in \text{Cl}(A_1)$, by (11) we have $A_2 \sim B$ for every minimum cut $B \in \text{MinCut}(A'_1 \cup A_2)$. With a slight abuse of notation, denote this by $\text{MinCut}(A'_1 \cup A_2) \sim A_2$.

Let CUT be a common minimum cut of the wiretap sets in $\text{Cl}(A_2)$. It follows that $\text{CUT} \sim A_2 \sim \text{MinCut}(A'_1 \cup A_2)$ for every $A'_1 \in \text{Cl}(A_1)$. Now, for each $A'_1 \in \text{Cl}(A_1)$, choose $\text{CUT}_{A'_1} \in \text{MinCut}(A'_1 \cup A_2)$ arbitrarily. Then $\text{CUT} \sim \text{CUT}_{A'_1}$ for every $A'_1 \in \text{Cl}(A_1)$. By Proposition 2, we obtain

$$\text{mincut}\left(s, \text{CUT} \cup \bigcup_{A'_1 \in \text{Cl}(A_1)} \text{CUT}_{A'_1}\right) = \text{mincut}(s, \text{CUT}) = |A_2|,$$

which implies that there exists a common minimum cut CUT^* of cardinality equal to $|A_2|$ separating CUT and $\text{CUT}_{A'_1}$ for every $A'_1 \in \text{Cl}(A_1)$. Therefore, CUT^* is a common minimum cut of the wiretap sets in $\text{Cl}(A_2)$ which also separates each of the wiretap sets in $\text{Cl}(A_1)$, i.e., $\text{Cl}(A_1) \prec \text{Cl}(A_2)$ by Definition 4. Theorem 4 is proved.

APPENDIX B

PROOF OF LEMMA 6

Let $|A_1| = r_1$, $|A_2| = r_2$ and clearly $r_1 < r_2$ since $A_1 \prec A_2$. Since A_1 is regular and $\text{CUT}_1 \in \text{MinCut}(A_1)$, it follows that

$$|\text{CUT}_1| = \text{mincut}(s, A_1) = |A_1| = r_1. \quad (12)$$

Further since $A_1 \prec A_2$ and $\text{CUT}_{1,2} \in \text{MinCut}(A_1 \cup A_2)$, by Proposition 3 we have

$$|\text{CUT}_{1,2}| = \text{mincut}(s, A_1 \cup A_2) = \text{mincut}(s, A_2) = |A_2| = r_2. \quad (13)$$

To prove $\text{CUT}_1 \prec \text{CUT}_{1,2}$, by Proposition 3, it suffices to prove that

$$\text{mincut}(s, \text{CUT}_1 \cup \text{CUT}_{1,2}) = \text{mincut}(s, \text{CUT}_{1,2}) = r_2.$$

First note that

$$\text{mincut}(s, \text{CUT}_1 \cup \text{CUT}_{1,2}) \geq \text{mincut}(s, A_1 \cup A_2) = r_2, \quad (14)$$

where the inequality follows from the fact that a cut between s and $\text{CUT}_1 \cup \text{CUT}_{1,2}$ separates $A_1 \cup A_2$ from s . Hence, we only need to prove that $\text{mincut}(s, \text{CUT}_1 \cup \text{CUT}_{1,2}) \leq r_2$.

Let $\text{mincut}(s, \text{CUT}_1 \cup \text{CUT}_{1,2}) = r$. Then there exist r edge-disjoint paths from s to the edges in $\text{CUT}_1 \cup \text{CUT}_{1,2}$, say P_1, P_2, \dots, P_r , such that each path passes through exactly one edge in $\text{CUT}_1 \cup \text{CUT}_{1,2}$ as the last edge of the path. This is explained as follows. Since the r last edges of the r paths are included in $\text{CUT}_1 \cup \text{CUT}_{1,2}$, if one path of them passes through more than one edge in $\text{CUT}_1 \cup \text{CUT}_{1,2}$, we can replace the path by its subpath from s to the first edge on the path in $\text{CUT}_1 \cup \text{CUT}_{1,2}$, and

this new path is still edge-disjoint with the other $r - 1$ paths and it passes through exactly one edge in $\text{CUT}_1 \cup \text{CUT}_{1,2}$, i.e., the last edge of the new path.

Let a ($a \leq r_1$ by (12)) be the number of paths among the r edge-disjoint paths P_1, P_2, \dots, P_r such that their last edges are in $\text{CUT}_1 \setminus \text{CUT}_{1,2}$. Then for the remaining $r - a$ paths, the $r - a$ last edges of them are in $\text{CUT}_{1,2}$. Without loss of generality, assume the former a paths be P_1, P_2, \dots, P_a with the last edges being $e_1, e_2, \dots, e_a \in \text{CUT}_1 \setminus \text{CUT}_{1,2}$, respectively, and the latter $r - a$ paths be $P_{a+1}, P_{a+2}, \dots, P_r$ with the last edges being $e_{a+1}, e_{a+2}, \dots, e_r \in \text{CUT}_{1,2}$, respectively. Let

$$\text{In}(\text{CUT}_1) = \{e_1, e_2, \dots, e_a\} \subseteq \text{CUT}_1 \setminus \text{CUT}_{1,2},$$

$$\text{In}(\text{CUT}_{1,2}) = \{e_{a+1}, e_{a+2}, \dots, e_r\} \subseteq \text{CUT}_{1,2},$$

and

$$\overline{\text{In}(\text{CUT}_{1,2})} = \text{CUT}_{1,2} \setminus \text{In}(\text{CUT}_{1,2}).$$

Then

$$|\text{In}(\text{CUT}_1)| + |\text{In}(\text{CUT}_{1,2})| = r. \quad (15)$$

Since CUT_1 is a minimum cut between s and the wiretap set A_1 , i.e., $|\text{CUT}_1| = \text{mincut}(s, A_1) = |A_1|$, there are r_1 edge-disjoint paths from CUT_1 to A_1 that start with all the r_1 distinct edges in CUT_1 and end with all the r_1 distinct edges in A_1 . Denote such r_1 paths by $P'_1, P'_2, \dots, P'_{r_1}$ and without loss of generality assume that P'_1, P'_2, \dots, P'_a start with e_1, e_2, \dots, e_a , respectively. Note that $P_i \cap P'_i = \{e_i\}$ for all $1 \leq i \leq a$, since the network G is acyclic. Next, we prove by contradiction that

$$P'_i \cap \text{CUT}_{1,2} \neq \emptyset, \quad \forall 1 \leq i \leq a. \quad (16)$$

Assume $P'_i \cap \text{CUT}_{1,2} = \emptyset$ for some i , $1 \leq i \leq a$. Since the path P_i from s to e_i does not contain any edge in $(\text{CUT}_1 \cup \text{CUT}_{1,2}) \setminus \{e_i\}$, $P_i \cup P'_i$ constitutes a path from s to some edge in A_1 not including any edge in $\text{CUT}_{1,2}$, which contradicts to the assumption that $\text{CUT}_{1,2}$ separates A_1 from s . Hence, we have proved (16).

We further prove by contradiction that

$$P'_i \cap \overline{\text{In}(\text{CUT}_{1,2})} \neq \emptyset, \quad \forall 1 \leq i \leq a. \quad (17)$$

Suppose $P'_i \cap \overline{\text{In}(\text{CUT}_{1,2})} = \emptyset$ for some i , $1 \leq i \leq a$. Note that P'_i does not pass through any edge in $\text{CUT}_1 \setminus \{e_i\}$. Together with (16), P'_i must pass through an edge in $\text{In}(\text{CUT}_{1,2}) \setminus \text{CUT}_1$. Consider the

last edge in $\text{In}(\text{CUT}_{1,2}) \setminus \text{CUT}_1$ that P'_i pass through. Without loss of generality, let this edge be e_{a+1} . Then

$$e_{a+1} \notin \text{CUT}_1. \quad (18)$$

Thus, the subpath of P'_i from e_{a+1} to some edge in A_1 does not contain any edge in $(\text{CUT}_1 \cup \text{CUT}_{1,2}) \setminus \{e_{a+1}\}$. Recall from the foregoing that the path P_{a+1} does not contain any edge in $(\text{CUT}_1 \cup \text{CUT}_{1,2}) \setminus \{e_{a+1}\}$. Then together with (18), we see that concatenating P_{a+1} and the subpath of P'_i from e_{a+1} to some edge in A_1 yields a path from s to some edge in A_1 without passing through any edge in CUT_1 . This contradicts the assumption that CUT_1 is a minimum cut between s and A_1 . Hence, we have proved (17).

Now, P'_i , $1 \leq i \leq a$, are edge-disjoint. Together with (17), we have $|\overline{\text{In}(\text{CUT}_{1,2})}| \geq a$, or equivalently, $|\text{In}(\text{CUT}_{1,2})| \leq r_2 - a$. It then follows from $|\text{In}(\text{CUT}_1)| = a$ and (15) that

$$r = |\text{In}(\text{CUT}_1)| + |\text{In}(\text{CUT}_{1,2})| \leq a + r_2 - a = r_2,$$

that is, $\text{mincut}(s, \text{CUT}_1 \cup \text{CUT}_{1,2}) \leq r_2$. Lemma 6 is proved.

APPENDIX C

PROOFS OF LEMMA 11 AND THEOREM 12

Proof of Lemma 11: The “only if” part of the lemma is trivial.

We now prove the “if” part. Let $\text{CUT}_1 = \{e_{1,i} : i = 1, \dots, n\}$ and $\text{CUT}_2 = \{e_{2,i} : i = 1, \dots, n\}$ be two minimum cuts in $\text{MinCut}(t)$. Let P_1, P_2, \dots, P_n be n edge-disjoint paths from s to t such that for each i , $1 \leq i \leq n$, $P_i \cap \text{CUT}_1 = \{e_{1,i}\}$, $P_i \cap \text{CUT}_2 = \{e_{2,i}\}$, and $e_{1,i} \leq e_{2,i}$. We now prove the “if” part by contradiction. Assume the contrary that $\text{CUT}_1 \leq \text{CUT}_2$ is false, i.e., CUT_1 is not a cut separating CUT_2 from s . Upon deleting the edges in CUT_1 , there still exists a path, say P , from s to an edge in CUT_2 , say $e_{2,1}$ (P includes $e_{2,1}$). Note that the path P and the subpath of P_1 from $\text{head}(e_{2,1})$ to t are edge-disjoint since the network is acyclic. In addition, the subpath of P_1 from $\text{head}(e_{2,1})$ to t does not contain the edges in CUT_1 since $P_1 \cap \text{CUT}_1 = \{e_{1,1}\}$ and $e_{1,1} \leq e_{2,1}$. Hence, concatenating P and the subpath of P_1 from $\text{head}(e_{2,1})$ to t yields a new path from s to t that contains no edges in CUT_1 , which contradicts the assumption that $\text{CUT}_1 \in \text{MinCut}(t)$. The proof is completed. ■

Proof of Theorem 12: We will prove the theorem by contradiction. Suppose that there exist n edge-disjoint paths from s to t , denoted by P_1, P_2, \dots, P_n , such that one of them, say P_1 , passes through an edge $e_{1,1} \in \text{CUT}_1$ and an edge $e_{2,1} \in \text{CUT}_2$ (i.e., $P_1 \cap \text{CUT}_1 = \{e_{1,1}\}$ and $P_1 \cap \text{CUT}_2 = \{e_{2,1}\}$) with $e_{2,1} < e_{1,1}$ (i.e., $e_{2,1} \leq e_{1,1}$ and $e_{2,1} \neq e_{1,1}$). Now, we divide the path P_1 into two disjoint subpaths: the

subpath from s to $e_{2,1}$ (including $e_{2,1}$), and the subpath from $\text{head}(e_{2,1})$ to t passing through $e_{1,1}$. By Proposition 10, the first subpath of P_1 from s to $e_{2,1}$ contains no edges in $(\text{CUT}_1 \cup \text{CUT}_2) \setminus \{e_{2,1}\}$. In other words, there exists a path from s to an edge in CUT_2 (i.e., $e_{2,1}$) upon deleting all the edges in CUT_1 , a contradiction to $\text{CUT}_1 \leq \text{CUT}_2$. The theorem is proved. \blacksquare

APPENDIX D

VERIFICATION OF ALGORITHM 3

In this appendix, we verify that the output edge set CUT of Algorithm 3 is the primary minimum cut between s and t . We adopt the standard terminologies in network flow theory. In a network G with a flow f , a non-source node u is called *reachable* from s if there exists an *f -unsaturated path* from s to u , where an *f -unsaturated path* means that each edge e on this path is either a forward edge with flow value 0 or a reverse edge with flow value 1. For a detailed discussion on unsaturated path, we refer the reader to [28, Chapter 7]. The following lemma is also standard.

Lemma D.1: *In a network G with a flow f from the source node s to a sink node t , if there exists an f -unsaturated path from s to t , then by “flipping” this path, i.e., replacing the flow value 0 of the forward edges on the path by 1 and the flow value 1 of the reverse edges on the path by 0, a new flow f' is obtained and the flow value of f' is increased by 1, i.e., $v(f') = v(f) + 1$. In particular, if no unsaturated paths from s to t exist, the flow is a maximum flow from s to t .*

Let $\text{CUT} = \{e_i : 1 \leq i \leq n\}$ be the output edge set of Algorithm 3. Then the nodes $\text{tail}(e_i)$, $1 \leq i \leq n$ are reachable and the nodes $\text{head}(e_i)$, $1 \leq i \leq n$ are unreachable from s . This further implies that all the edges in CUT have flow value 1, i.e., $f(e_i) = 1$, $1 \leq i \leq n$, because otherwise $\text{head}(e_i)$ would be included in the set S when the algorithm terminates.

First, we can easily see that CUT is indeed a cut between s and t , i.e., $t \notin S$, because otherwise there exists an unsaturated path from s and t , implying that f is not a maximum flow by Lemma D.1. It follows that $n \geq C_t \geq 1$.

We now prove that CUT is minimum, i.e., $n = C_t$. Assume the contrary that $n > C_t$. Then the maximum flow f can be decomposed into C_t edge-disjoint paths P_1, P_2, \dots, P_{C_t} from s to t with

$$f(e) = \begin{cases} 1, & e \in P_i \text{ for some } 1 \leq i \leq C_t; \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

Since $f(e_i) = 1$ for $1 \leq i \leq n$, each e_i must be on one of the C_t edge-disjoint paths from s to t . Furthermore, since $n > C_t$, there exists a path P_j that contains at least 2 edges in CUT, say e_1 and e_2 . We assume without loss of generality that $e_1 \leq e_2$ on P_j . Note that $\text{tail}(e_2)$ is reachable from s . If

$\text{tail}(e_2) = \text{head}(e_1)$, then $\text{head}(e_1)$ is reachable from s , which is a contradiction because $e_1 \in \text{CUT}$. Otherwise, let \hat{e} be the predecessor of e_2 on P_j . Since $f(\hat{e}) = 1$, $\text{tail}(\hat{e})$ is also reachable from s (through $\text{tail}(e_2)$). By repeating this argument if necessary, we see inductively that $\text{head}(e_1)$ is reachable from s , a contradiction. Therefore, CUT must be a minimum cut between s and t , i.e., $n = C_t = v(f)$.

It remains to prove that CUT is primary. Assume that CUT is not primary, and instead let $\text{CUT}^* = \{e_i^* : 1 \leq i \leq n\}$ be the primary minimum cut between s and t . By Definitions 6 and 7, we have $\text{CUT}^* \leq \text{CUT}$. By Proposition 10, we can let $\text{CUT} \cap P_i = \{e_i\}$ and $\text{CUT}^* \cap P_i = \{e_i^*\}$ for $1 \leq i \leq n$. Then $e_i^* \leq e_i$ for all $1 \leq i \leq n$ by Theorem 12, which implies that for each $1 \leq i \leq n$, the subpath of P_i from $\text{head}(e_i)$ to t contains no edges in CUT^* . Since we assume that $\text{CUT} \neq \text{CUT}^*$, there exists $1 \leq i \leq n$ such that $e_i \neq e_i^*$. Without loss of generality assume that $e_1 \neq e_1^*$, and let P be an f -unsaturated path from s to $\text{tail}(e_1)$.

Now, consider any edge $e \in P \cap P_1$. Since $e \in P_1$, we have $f(e) = 1$, which together with $e \in P$ implies that e must be a reverse edge on P . Thus, we have proved the following claim which will be used throughout the rest of the proof.

Claim 1: *For every edge $e \in P \cap P_1$, $\text{head}(e)$ is the node on P immediately before $\text{tail}(e)$.*

We now prove by contradiction that P and the subpath of P_1 from $\text{tail}(e_1)$ to t , denoted by $P_1^{\text{tail}(e_1) \rightarrow t}$,⁷ are edge-disjoint. Let $e \in P \cap P_1^{\text{tail}(e_1) \rightarrow t}$. We consider two cases:

$e = e_1$ By Claim 1, we can see that $\text{tail}(e_1)$ is reachable from s on P through $\text{head}(e_1)$, which implies that $\text{head}(e_1)$ is also reachable from s , a contradiction.

$e \neq e_1$ Since $e \in P \cap P_1^{\text{head}(e_1) \rightarrow t}$, $\text{tail}(e)$ is reachable from s (through $\text{head}(e)$) since e is on P . Together with the flow value of each edge (if exists) on the subpath $P_1^{\text{head}(e_1) \rightarrow \text{tail}(e)}$ being 1, by the argument previously used in proving that CUT is minimum, $\text{head}(e_1)$ is also reachable from s , which again is a contradiction.

We now prove by contradiction that CUT^* is not the primary minimum cut between s and t by considering two cases.

Case 1: $P \cap P_1^{s \rightarrow \text{tail}(e_1)} = \emptyset$.

⁷Let P be an (unsaturated) path from s to a non-source node u . For any two nodes u_1 preceding u_2 on P , the subpath of P from u_1 to u_2 is denoted by $P^{u_1 \rightarrow u_2}$ throughout this proof to simplify notation.

We will prove that in this case a new maximum flow f'' with $f''(e_1^*) = 0$ can be found, i.e., the n edge-disjoint paths from s to t with respect to f'' does not pass through e_1^* . First, we define f' as

$$f'(e) = \begin{cases} 0, & e \in P_1; \\ f(e), & \text{otherwise;} \end{cases}$$

which is a flow but no longer a maximum flow since $v(f') = v(f) - 1 = n - 1$. Then we can obtain an f' -unsaturated path \hat{P}_1 from s to t by concatenating P and $P_1^{\text{tail}(e_1) \rightarrow t}$. Since $e_1^* < e_1$ on P_1 , e_1^* is on the subpath $P_1^{s \rightarrow \text{tail}(e_1)}$. Together with $P \cap P_1^{s \rightarrow \text{tail}(e_1)} = \emptyset$, we have $e_1^* \notin \hat{P}_1$. Now, define a flow f'' by flipping the flow values in the f' -unsaturated path \hat{P}_1 , i.e.,

$$f''(e) = \begin{cases} 1, & e \in \hat{P}_1 \text{ with } f'(e) = 0; \\ 0, & e \in \hat{P}_1 \text{ with } f'(e) = 1; \\ f'(e), & \text{otherwise.} \end{cases} \quad (20)$$

By Lemma D.1, we have $v(f'') = v(f') + 1 = n$. We then have obtained a maximum flow f'' with $f''(e_1^*) = 0$. By Proposition 10, CUT^* is not a minimum cut between s and t , and hence not the (primary) minimum cut between s and t .

Case 2: $P \cap P_1^{s \rightarrow \text{tail}(e_1)} \neq \emptyset$.

Let e' be the first edge on P that is also on P_1 . We consider two cases.

Case 2A: $e_1^* \leq e'$.

By Claim 1, e' is a reverse edge on P with $f(e') = 1$ and $e' < e_1$. Then the f -unsaturated subpath $P^{s \rightarrow \text{head}(e')}$ does not pass through $\text{tail}(e')$, and hence does not contain e' . On the other hand, since e' is the first edge on P that is also on P_1 , $P^{s \rightarrow \text{head}(e')}$ does not contain any edge on P_1 . Thus, $P^{s \rightarrow \text{head}(e')}$ is edge-disjoint with P_1 , and therefore also with the subpath $P_1^{s \rightarrow \text{head}(e')}$. Since $e_1^* \leq e'$, e_1^* is on $P_1^{s \rightarrow \text{head}(e')}$ and hence not on $P^{s \rightarrow \text{head}(e')}$ and $P_1^{\text{head}(e') \rightarrow t}$. By considering the concatenation of $P^{s \rightarrow \text{head}(e')}$ and $P_1^{\text{head}(e') \rightarrow t}$, we see by using the same argument as in Case 1 that CUT^* is not the primary minimum cut between s and t .

Case 2B: $e_1^* > e'$.

Let \tilde{e} be the last edge on $P \cap P_1$ such that $\tilde{e} < e_1^*$. Consider the following two cases:

- 1) $P^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)} \cap P_1^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)} = \emptyset$. Since $\tilde{e} < e_1^* < e_1$, $e_1^* \in P_1^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)}$ and hence $P^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)}$ does not contain e_1^* . On the other hand, since \tilde{e} is the last edge on $P \cap P_1$ such that $\tilde{e} < e_1^*$, $P^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)}$ contains no edges on P_1 , where we note that \tilde{e} is not on $P^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)}$ by Claim 1. Then $P^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)}$ is edge-disjoint with P_1 . By considering the concatenation of $P_1^{s \rightarrow \text{tail}(\tilde{e})}$, $P^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)}$, and $P_1^{\text{tail}(e_1) \rightarrow t}$, we see by using the same argument as in Case 1 that CUT^* is not the primary minimum cut between s and t .

- 2) $P^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)} \cap P_1^{\text{tail}(\tilde{e}) \rightarrow \text{tail}(e_1)} \neq \emptyset$. Let \hat{e} be the first edge on $P \cap P_1$ such that $\hat{e} \geq e_1^*$. Together with \tilde{e} being the last edge on $P \cap P_1$ such that $\tilde{e} < e_1^*$, the subpath $P^{\text{tail}(\tilde{e}) \rightarrow \text{head}(\hat{e})}$ contains no edges on P_1 by Claim 1, and hence $P^{\text{tail}(\tilde{e}) \rightarrow \text{head}(\hat{e})}$ is edge-disjoint with P_1 . On the other hand, we note that $\tilde{e} < e_1^* \leq \hat{e} < e_1$ on P_1 , implying that $e_1^* \in P_1^{\text{tail}(\tilde{e}) \rightarrow \text{head}(\hat{e})}$. Thus, considering the concatenation of $P_1^{s \rightarrow \text{tail}(\tilde{e})}$, $P^{\text{tail}(\tilde{e}) \rightarrow \text{head}(\hat{e})}$, and $P_1^{\text{head}(\hat{e}) \rightarrow t}$, we see by using the same argument as in Case 1 that CUT* is not the primary minimum cut between s and t .

Combining all the above, Algorithm 3 is verified.

REFERENCES

- [1] C. E. Shannon, "Communication theory of secrecy systems," *Bell Sys. Tech. J.*, vol. 28, pp. 656-715, 1949.
- [2] G. R. Blakley, "Safeguarding cryptographic keys," in *Proc. National Computer Conference*, 1979, vol. 48, pp. 313-317.
- [3] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, pp. 612-613, 1979.
- [4] L. H. Ozarow and A. D. Wyner, "Wire-tap channel II," *AT&T Bell Labs. Tech. J.*, vol. 63, pp. 2135-2157, 1984.
- [5] M. Celebiler, G. Stette, "On increasing the down-link capacity of a regenerative satellite repeater in point-to-point communications," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 98-100, Jan. 1978.
- [6] R. W. Yeung and Z. Zhang, "Distributed source coding for satellite communications," *IEEE Trans. Inf. Theory*, vol. 45, no. 4, pp. 1111-1120, May 1999.
- [7] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204-1216, Jul. 2000.
- [8] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371-381, Jul. 2003.
- [9] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782-795, Oct. 2003.
- [10] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. G. M. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inf. Theory*, vol. 51, no. 6, pp. 1973-1982, Jun. 2005.
- [11] M. Tan, R. W. Yeung, S.-T. Ho, and N. Cai, "A unified framework for linear network coding," *IEEE Trans. Inf. Theory*, vol. 57, no. 1, pp. 416-423, Jan. 2011.
- [12] Q. Sun, S. T. Ho, and S.-Y. R. Li, "Network matroids and linear network codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Toronto, ON, Canada, Jul. 2008, pp. 1833-1837.
- [13] Q. Sun, X. Yin, Z. Li, K. Long, "Multicast network coding and field sizes," *IEEE Trans. Inf. Theory*, vol. 61, no. 11, pp. 6182-6191, Nov. 2015.
- [14] R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang, "Network coding theory," *Foundations and Trends in Communications and Information Theory*, vol. 2, nos.4 and 5, pp. 241-381, 2005.
- [15] R. W. Yeung, *Information Theory and Network Coding*. New York: Springer, 2008.
- [16] C. Fragouli and E. Soljanin, "Network coding fundamentals," *Foundations and Trends in Networking*, vol. 2, no.1, pp. 1-133, 2007.
- [17] C. Fragouli and E. Soljanin, "Network coding applications," *Foundations and Trends in Networking*, vol. 2, no.2, pp. 135-269, 2007.

- [18] T. Ho and D. S. Lun, *Network Coding: An Introduction*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
- [19] N. Cai and R. W. Yeung, "Secure network coding," *IEEE Int. Symp. Inf. Theory*, Lausanne, Switzerland, Jun. 30-Jul. 5, 2002.
- [20] N. Cai and R. W. Yeung, "Secure Network Coding on a Wiretap Network," *IEEE Trans. Inf. Theory*, vol. 57, no. 1, pp. 424-435, Jan. 2011.
- [21] S. El Rouayheb, E. Soljanin, and A. Sprintson, "Secure Network Coding for Wiretap Networks of Type II," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1361-1371, March 2012.
- [22] D. Silva and F. R. Kschischang, "Universal secure network coding via rank-metric codes," *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 1124-1135, Feb. 2011.
- [23] J. Feldman, T. Malkin, R. A. Servedio, and C. Stein, "On the capacity of secure network coding," 42nd Ann. Allerton Conf. Commun., Contr., Comput., Monticello, IL, Sep. 29-Oct. 1, 2004.
- [24] X. Guang, J. Lu, and F.-W. Fu, "Small field size for secure network coding," *IEEE Commun. Lett.*, vol. 19, no. 3, pp. 375-378, March 2015.
- [25] L. R. Ford Jr. and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, no. 3, pp. 399-404, 1956.
- [26] P. Elias, A. Feinstein, and C. E. Shannon, "A note on maximum flow through a network," *IRE Trans. Inf. Theory*, col. 2, vol. 4, pp. 117-119, April 1956.
- [27] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
- [28] J. A. Bondy and U. S. R. Murty, *Graph Theory*. Springer, 2008.
- [29] R. K. Ahuja and J. B. Orlin, "Distance-directed augmenting path algorithms for maximum flow and parametric maximum flow problems," *Naval Research Logistics (NRL)*, vol. 38, no. 3, pp. 413-430, June 1991.
- [30] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Soviet Mathematics Doklady*, vol. 11, no. 5, pp. 1277-1280, 1970.